Fast Screening Algorithm for Template Matching

FAST SCREENING ALGORITHM FOR TEMPLATE MATCHING

BY

BOLIN LIU, B.Sc.

A THESIS

SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

AND THE SCHOOL OF GRADUATE STUDIES

OF MCMASTER UNIVERSITY

IN PARTIAL FULFILMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF APPLIED SCIENCE

Master of Applied Science (2017) McMaster University

(Electrical & Computer Engineering) Hamilton, Ontario, Canada

TITLE:	Fast Screening Algorithm for Template Matching
AUTHOR:	Bolin Liu B.Sc., (Communication Engineering) Beijing Institute of Technology, Beijing, China
SUPERVISOR:	Dr. Xiaolin Wu
NUMBER OF PAGES:	xii, 70

To my family and friends

Abstract

This paper presents a generic pre-processor for expediting conventional template matching techniques. Instead of locating the best matched patch in the reference image to a query template via exhaustive search, the proposed algorithm rules out regions with no possible matches with minimum computational efforts. While working on simple patch features, such as mean, variance and gradient, the fast pre-screening is highly discriminative. Its computational efficiency is gained by using a novel octagonal-star-shaped template and the inclusion-exclusion principle to extract and compare patch features. Moreover, it can handle arbitrary rotation and scaling of reference images effectively, and also be robust to uniform illumination changes. GPU-aided implementation shows great efficiency of parallel computing in the algorithm design, and extensive experiments demonstrate that the proposed algorithm greatly reduces the search space while never missing the best match.

Acknowledgements

First of all, I would like to take this opportunity to express my deepest and sincerest gratitude to my supervisor, Prof. Xiaolin Wu, for his consistent and invaluable guidance and strong support throughout my Master program. He not only provided me with in-depth knowledge, but also inspired me with the spirit of exploring science and seeking truth. His kind encouragement and keen insights are highly appreciated and will always be remembered and cherished. This work would have not been possible without his help.

Secondly, I want to give my special thanks to Dr. Xiao Shu, for our many discussions and his valuable advices, which certainly helped me to stay at the right track. His support is also indispensable in this work.

Furthermore, I would like to thank Dr. Sorina Dumitrescu and Dr. Jiankang Zhang for being members in my defence committee. I appreciate their time reviewing my thesis and providing valuable feedback.

Last but not least, I am indebted to my parents for their love and support. They made great efforts and sacrificed a lot in order to help me concentrate on my academic career and complete my chosen path. To them, I dedicate this thesis.

Notation and abbreviations

SAD	Sum of Absolute Difference
SSD	Sum of Squared Difference
FFT	Fast Fourier Transform
NCC	Normalized Cross Correlation
FS	Full Search
SIFT	Scale Invariant Feature Transform
ASIFT	Affine-SIFT
DOG	Difference of Gaussian
PCA	Principal Components Analysis
GLOH	Gradient Location-Orientation Histogram
SURF	Speeded-Up Robust Features
LDE	Linear Discriminant Embedding
BBS	Best-Buddies Similarity

BBP	Best-Buddies Pair
CenSurE	Centre Surround Extrema
CUDA	Computer Unified Device Architecture
SIMD	Single Instruction Multiple Data
SM	Streaming Multiprocessor

Contents

List of Figures

Chapter 1

Introduction and Problem Statement

As a subclass of general image matching, template matching is one of the most fundamental task in image processing and computer vision. In past decades, it has been deeply and widely studied for diverse applications, such as image and video compression, motion estimation, image restoration, object detection and tracking. As shown in Figure 1.1, template matching is to locate the best matched patch in a reference image to a given query template. While finding an object in image seems to be trivial for human in general, it is a surprisingly challenging problem for computer algorithms. Most existing template matching techniques are very time-consuming and demand a lot of computational resources to obtain relatively reliable results in real-world applications, where geometric distortions between the reference image and query template are unavoidable.

To handle geometric distortions like a simple change in scale or orientation, conventional patch based template matching techniques, which compare the template directly with all the candidate patches in succession, have to rely on an exhaustive search of all the combinations of different scales and rotations. In comparison, newly emerged feature based techniques are more efficient and robust for matching a template with deformations. The

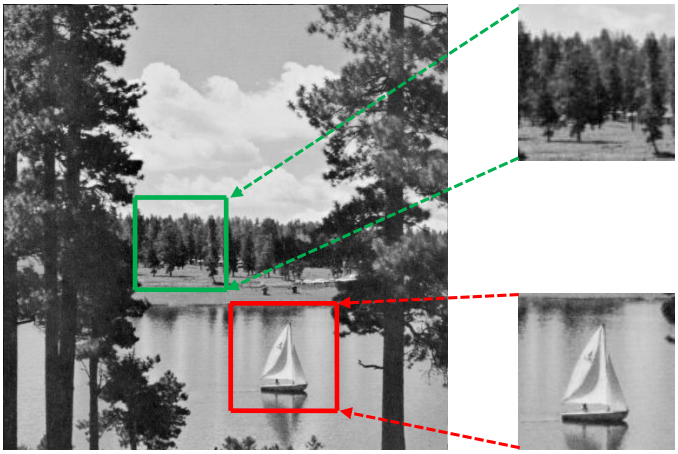


Figure 1.1: Example of templates and reference image.

basic idea of feature based template matching is to generate some feature points from the query template and check if these feature points also occur in some patches in the reference image. As matched points do not need to appear at the exact matched positions, feature based techniques are more resilient to geometric distortions as long as both of the reference image and query template have distinctive features. However, with regard to overall matching speed, feature based techniques are still unsatisfactory; they commonly require a computationally intensive process to generate features for different scales, and the whole matching process can be slow for high-resolution images.

In this paper, we propose a rotation, scale and illumination invariant template matching pre-processor for expediting conventional patch and feature based template matching techniques. Instead of pinpointing the best matched patch, this pre-processor rapidly rules out regions of no possible matches using simple patch features, such as mean, variance and gradient. After the pre-processing procedure, the remainder parts of the reference image are passed to a more accurate but slower template matching technique to locate the exact position of the best match. As the regions to search are greatly reduced in the pre-processing

step, the accurate template matching technique needs to process far fewer pixels or patches hence it runs much faster than having to process the whole image.

To make this two-stage method a viable strategy, the cost of the pre-processing, i.e., the time spent on identifying no match regions, must be less than the time saved in the later accurate template matching stage. Thus, the major challenge in the design of the pre-processing algorithm is to make its computational complexity extremely low while effective enough to mark as many unmatched regions as possible. On the other hand, since the accurate template matching technique in the second stage only operates on the unmarked regions from the first stage, if the best matched patch has already been incorrectly marked as unmatched, the two-stage method would fail to locate the best match. Therefore, in order to avoid affecting the success rate of the subsequent template matching, the false positive rate of the pre-processing algorithm must be made very low.

The remainder of this paper is organized as follows. Chapter 2 reviews some of the related template matching techniques. Chapter 3 discusses the patch features employed by the proposed technique, and elaborates on how to make the proposed technique rotation, scale and illumination invariant, respectively. Chapter 4 shows the parallel implementation of the proposed algorithm and compares the performance based on CPU and GPU. Chapter 5 presents extensive experimental results on image matching and video tracking as pre-processor. Chapter 6 concludes and discusses some future work.

Chapter 2

Background and Related Work

A great amount of research effort has been dedicated to designing efficient and effective template matching techniques in past decades. One important difference of various matching techniques is the similarity measure between the given query template and patches in the reference image. Utilizing different features or similarity functions, existing template matching techniques could be divided into two classes: patch based techniques and feature based techniques. The former class of template matching techniques establishes similarity comparison based on the statistical features of a whole patch, while the latter one locates the best match with the help of local feature points.

2.1 Patch Based Techniques

The most basic similarity measures for patch based techniques are Sum of Absolute Difference (SAD) and Sum of Squared Difference (SSD). Suppose that we are looking for an $n_1 \times n_2$ template T in an $M_1 \times M_2$ reference image I. An intuitive way is to compare the template with each candidate patch of the same size in the image with some similarity

measures. In the following, we use vectors X_t and X_p of length $N = n_1 \times n_2$ to represent the template and a candidate patch in the image, respectively.

There are several ways to compute the similarity of two vectors. One of the most commonly used methods is the L_p-norm, which is defined for vector $Z = [z_0, z_1, z_2, ..., z_{N-1}]$ as follows,

$$||Z||_p = (|z_0|^p + |z_1|^p + |z_2|^p + \cdots + |z_{N-1}|^p)^{1/p}. \tag{2.1}$$

Based on the L_p-norm, the similarity between the template and a candidate patch can be measured by the distance of vectors X_t and X_p as follows.

$$d(X_t, X_p) = ||X_t - X_p||_p^p. \tag{2.2}$$

If $p = 1$, the distance measure is commonly referred as the sum of absolute difference (SAD), while $p = 2$, the measure yields the sum of squared difference (SSD). The smaller $d(X_t, X_p)$ is, the more similar X_t and X_p are.

SAD and SSD are widely adopted due to their simplicity. However, Full Search (FS) algorithm in a whole image, which means comparing template with all equally sized sliding windows using same similarity function, will still be unacceptably slow with SAD or SSD. In this sense, many Full Search equivalent (FS-equivalent) methods, which reject many candidate windows by using some simpler similarity measures rather than carry heavy computation everywhere, have been proposed in previous work. On the topic of the latest FS-equivalent techniques, Ouyang *et al.* provided a comprehensive survey and compared the performances of several popular methods on their efficiency and robustness (Ouyang *et al.*, 2012). According to their evaluation results, the best performing FS-equivalent method

is the Low-Resolution Pruning algorithm (Gharavi-Alkhansari, 2001), which reduces the search space by pruning unmatched regions using a down-sampled reference image. Besides this coarse-to-fine strategy, another effective approach to speed-up the FS algorithm is to optimize the step size of sliding windows in reference images. The rank measure algorithm proposed by Pele and Werman is an example of this idea (Pele and Werman, 2007).

When the query template is large, SSD computation can be greatly accelerated by the Fast Fourier Transform (FFT) based approach. For a reference image of length $M = M_1 \times M_2$, a query template of N pixels requires $O(MN)$ time with brute force FS algorithm, while FFT-based approach needs only $O(M \log_2 M)$ time. If N is larger than $\log_2 M$, FFT-based approach would be faster. The idea is to rewrite SSD function as,

$$||X_t - X_p||_2^2 = ||X_t||_2^2 + ||X_p||_2^2 - 2 \cdot \langle X_t, X_p \rangle. \tag{2.3}$$

where $\langle X_t, X_p \rangle$ represents the inner product of X_t and X_p. FFT reduces computation of inner product by transforming complex spatial domain convolution to pixel-wise multiplication in frequency domain (Brown, 1992). The L_2 norm $||X_p||_2^2$ of each patch in the reference image could be obtained efficiently using the summed area table, which is also referred as the integral image (Crow, 1984; Lewis, 1995; Viola and Jones, 2004). We discuss this in detail in Section 3.1.

In addition to SAD and SSD, another popular similarity measure is Normalized Cross Correlation (NCC), which is designed specifically to be robust against uniform illumination

change. The definition of NCC is given as follows,

$$NCC(X_t, X_p) = \frac{\langle (X_t - \overline{X}_t \cdot \mathbf{1}), (X_p - \overline{X}_p \cdot \mathbf{1}) \rangle}{||X_t - \overline{X}_t||_2 \cdot ||X_p - \overline{X}_p||_2} \tag{2.4}$$

where $\mathbf{1}$ is a all-ones vector of length N, and $\overline{X}_t$ and $\overline{X}_p$ are the mean value of the vector X_t and X_p, respectively.

Similar to SSD, template matching techniques based on NCC can also be accelerated using frequency domain approaches and summed area tables.

Expand Eq. (2.4) to be,

$$\begin{aligned} & NCC(X_t, X_p) \\ =& \frac{\langle X_t, X_p \rangle - \overline{X}_t \cdot \sum_{i=0}^{N-1} X_{p_i} - \overline{X}_p \cdot \sum_{i=0}^{N-1} X_{t_i} + \sum_{i=0}^{N-1} (\overline{X}_t \cdot \overline{X}_p)}{\sqrt{\sum_{i=0}^{N-1} (X_{t_i} - \overline{X}_t)^2} \cdot \sqrt{\sum_{i=0}^{N-1} (X_{p_i} - \overline{X}_p)^2}} \\ =& \frac{\langle X_t, X_p \rangle - N \cdot \overline{X}_t \cdot \overline{X}_p}{\sqrt{\sum_{i=0}^{N-1} (X_{t_i} - \overline{X}_t)^2} \cdot \sqrt{\sum_{i=1}^{N-1} X_{p_i}^2 - N \cdot \overline{X}_p^2}}. \end{aligned} \tag{2.5}$$

where $N = n_1 \times n_2$ is the length of the given template vector. For the numerator of Eq. (2.5), FFT can speed-up computation of the inner product $\langle X_t, X_p \rangle$. $N \cdot \overline{X}_t$ is constant and $\overline{X}_p$ could be obtained by the summed area table. Additionally, for the denominator of Eq. (2.5), $\sqrt{\sum_{i=0}^{N-1} (X_{t_i} - \overline{X}_t)^2}$ is constant. With two different summed area tables, we could get $\sum_{i=1}^{N-1} X_{p_i}^2$ and $\overline{X}_p^2$ easily, respectively.

By taking the advantage of FFT, many SAD, SSD and NCC based techniques have highly efficient implementations, however, they are generally not flexible enough to handle geometric transformation between the reference image and query template effectively. There have been a few attempts to add rotation and scale invariant support in SAD, SSD and NCC based approaches. For instance, Ullah and Kaneko used the orientation code to

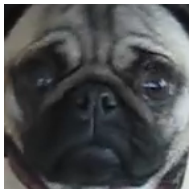
represent gradient information in a patch for approximating rotation angles as well as for rotation invariant matching (Ullah and Kaneko, 2004). Choi and Kim proposed to combine circular projection and Zernike moments to achieve rotation invariance (Choi and Kim, 2002). Their method was later improved by Kim (Kim, 2010). Based on circular projection, Kim used the Fourier coefficients of radial projections to be new rotation invariant and discriminating features and also handle the scale problem with pre-defined scale factors in an enumeration way. Lin and Chen used ring projection transform to establish parametric template vector for differently scaled template to get invariance for rotation and scale (Lin and Chen, 2008). Moreover, Tsai and Chiang solved the rotation problem using wavelet decompositions and ring projection (Tsai and Chiang, 2002). Wakahara and Yamashita proposed a global projection transform correlation method to deal with arbitrary 2D transformation (Wakahara and Yamashita, 2014).

Another recent work is Fast-Match (Korman *et al.*, 2013). Fast-Match can approximate the global optimum of any 2D affine transformation in reference image using a transformation net with SAD error bound δ if the input image and template are smooth. An example output of Fast-Match is shown in Figure 2.1.

Each estimated transformation in Fast-Match could be evaluated by the normalized SAD distance $\Delta_t(T, I)$ between the template and the reference image, where the $\Delta_t(T, I)$ is defined as,

$$\Delta_t(T, I) = \frac{1}{N} \sum_{p \in T} |T(p) - I(t(p))|. \tag{2.6}$$

The subscript t denotes the transformation and p represents a pixel. The goal of locating the best match is equivalent to finding the transformation t that minimizes the distance $\Delta_t(T, I)$.



(a)

(b)

Figure 2.1: Matching examples of Fast-Match. (a) is the given template. (b) shows the matching result in the reference image. Matching area is marked by red boxes.

In order to construct the transformation net efficiently, Fast-Match only samples a small number of transformations, and defined the l_∞ to measure the distance between two transformations, t_1 and t_2. Mathematically,

$$l_\infty(t_1, t_2) = \max_{p \in T} ||t_1(p) - t_2(p)||_2, \tag{2.7}$$

where $||\cdot||_2$ denotes the Euclidean distance of two pixels in the reference image. Thus, with the assumption that the template and reference image are smooth, the difference between $\Delta_{t_1}(T, I)$ and $\Delta_{t_2}(T, I)$ can be bounded in terms of l_∞. This enables sampling affine transformations to construct a limited transformation net to be possible. In Fast-Match, the error bound δ is defined as an input parameter for the constructed net $\mathbb{N}_\delta$, which means for any affine transformation t, there must exist some t_j in $\mathbb{N}_\delta$, such that $l_\infty(t, t_j) = \Theta(\delta)$. The size of $\mathbb{N}_\delta$ is $\Theta(\frac{1}{\theta^6} \cdot \frac{M}{N})$, where, N and M denote the number of pixels in the template and

reference image, respectively.

In order to speed up the searching and comparing in the constructed net, Fast-Match proposed a sub-linear algorithm for approximating the distance $\Delta_t(T, I)$ of a single transform. Instead of computing the distance for all the pixels, inspecting only a fraction of pixels can achieve a good estimation for the whole distance. As the net size grows rapidly with smaller precision parameter δ, achieving a satisfactory error rate becomes impractical. Therefore, Fast-Match also adopts a coarse-to-fine strategy to build and search through the transformation net.

The Achilles' heel of Fast-Match is that its runtime increases rapidly with the decrease of template size, hence inefficient for small templates. For example, when the height and width of the template is 10% of which of the reference image, the Fast-Match requires about tens of seconds according to the original authors, making our proposed pre-processing algorithm necessary.

2.2 Feature Based Techniques

Generally speaking, patch based techniques are inflexible and not robust against outliers, partial illumination changes and occlusions. Due to the limitations of patch-based techniques, there is a growing interest in feature-based techniques, such as Scale Invariant Feature Transform (SIFT) (Lowe, 2004). SIFT obtains rotation and scale invariant features for both template and reference image, and employs data fitting algorithms like RANSAC (Fischler and Bolles, 1981) to find matching patterns. An example output of SIFT is given in Figure 2.2.

SIFT obtains features that are invariant to scale changes in an image by searching

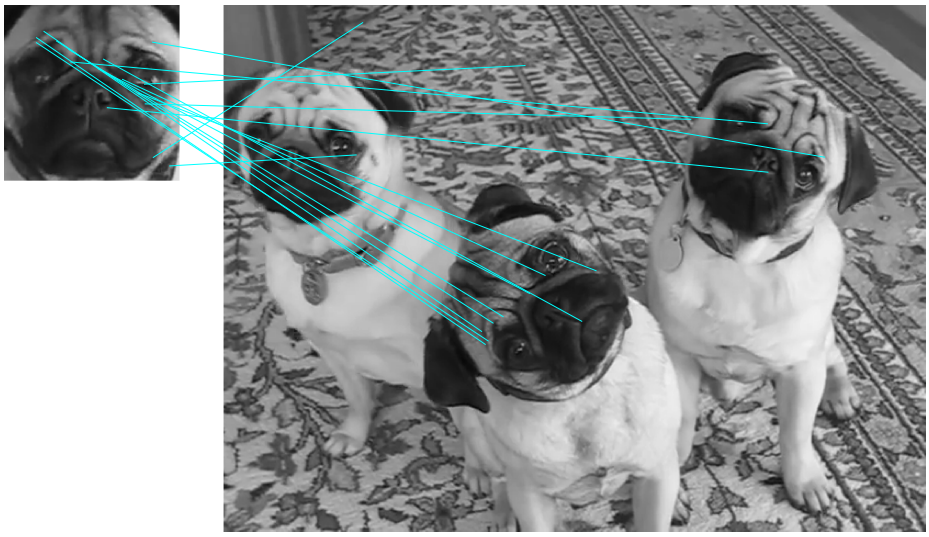


Figure 2.2: Matching examples of SIFT. Matched key points are linked with lines between the template (left side) and the reference image (right side).

for stable key points across all possible scales, with the well-known scale space function $L(x, y, \sigma)$ (Babaud *et al.*, 1986). $L(x, y, \sigma)$, also called Gaussian image, is generated from the convolution of a variable-scale Gaussian kernel $G(x, y, \sigma)$ and a reference image $I(x, y)$, as follows,

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \tag{2.8}$$

Then, the scale space extrema are computed from the Difference-of-Gaussian (DOG) function, which is the difference of two adjacent layers separated by a multiplicative factor k in a Gaussian images pyramid:

$$\begin{aligned} D(x, y, \sigma) =& (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ =& L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned} \tag{2.9}$$

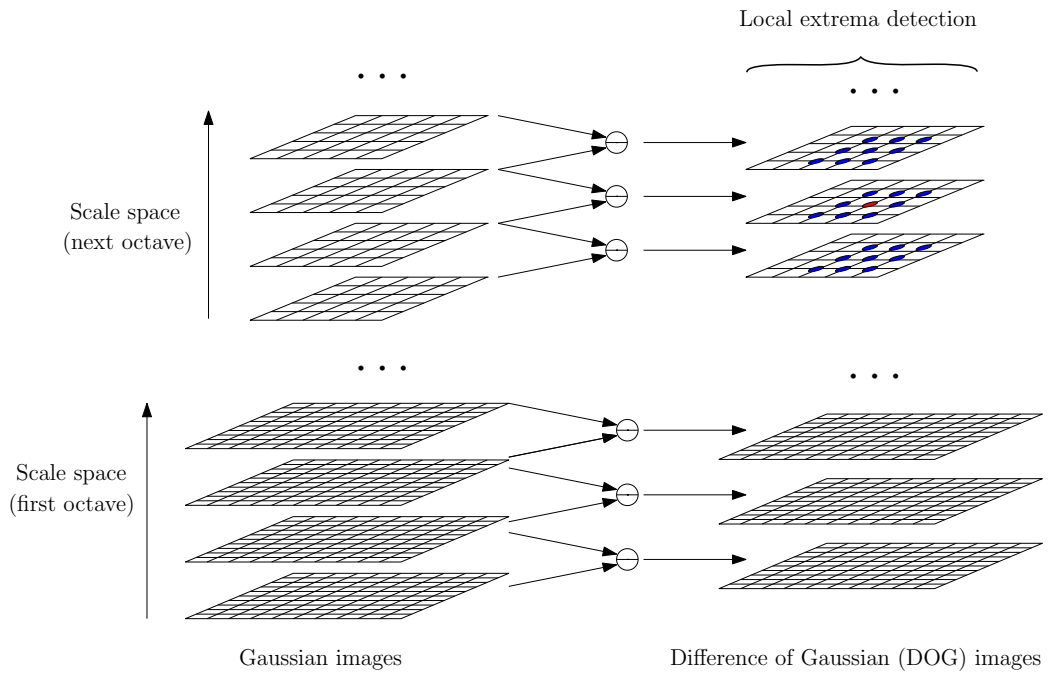


Figure 2.3: DOG images (right column) are generated by two nearby Gaussian images (left column) and scale space extrema are detected by comparing a pixel (marked with red spot) to its 26 neighbours in 3×3 regions at the current and adjacent scales (marked with blue spots).

The whole scale space of one reference image consists of several scale octaves, as shown in Figure 2.3. Key points are selected from all the extrema based on stability constraints.

The orientation is then assigned to each key point location based on local image gradient directions. In a Gaussian image, $L(x, y)$, the gradient magnitude and direction of a pixel are computed as,

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}, \tag{2.10}$$

$$\theta(x,y) = \tan^{-1}[(L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))]. \tag{2.11}$$

Within a region around the key point, SIFT quantizes the gradient direction of each pixel

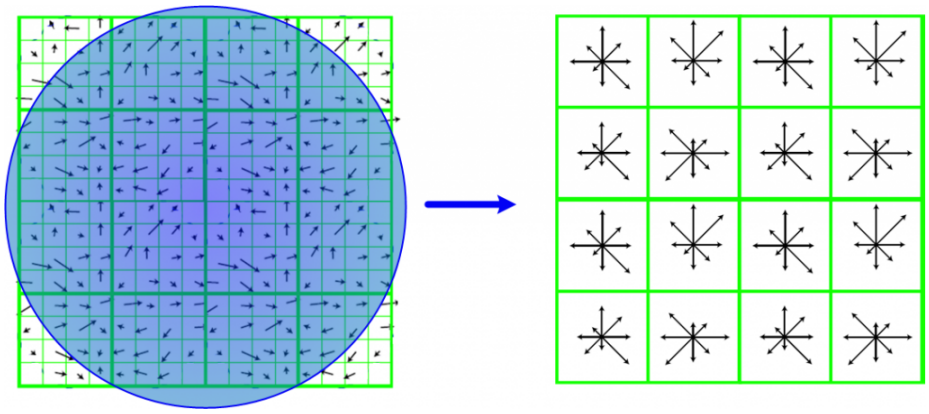


Figure 2.4: The key point descriptor is created by first computing gradient magnitude and direction for 16×16 sample points around its location. These magnitudes are also weighted by a Gaussian window, indicated by the overlaid circle. The final orientation histogram summarized the contents over 4×4 subregions, where each subregion covers 360 degrees with 8 bins. Thus, the descriptor includes $4 \times 4 \times 8 = 128$ elements.

into one of the 8 bins and adds the gradient magnitude into the corresponding bin. With the formed histogram, the bin with maximum value will be determined as the orientation of the key point.

In order to achieve rotation invariance, the coordinates of the local descriptor are rotated relative to the key point orientation to do the normalization. The feature vector is then formed with rotated local gradient information. Thus, with the location, scale and orientation, SIFT can generate a 128-element feature vector for each key point, as shown in Figure 2.4. Normalized feature vectors across the scale space help SIFT achieve its rotation and scale invariance.

Mikolajczyk and Schmid presented a comparative study of several local descriptors (Mikolajczyk and Schmid, 2005), showing that the SIFT descriptor is superior to many other descriptors, such as the distribution-based shape context (Belongie *et al.*, 2002), the geometric histogram descriptors (Ashbrook *et al.*, 1995), the derivative-based complex filters (Schaffalitzky and Zisserman, 2002), and the moment invariants (Van Gool *et al.*,

1996).

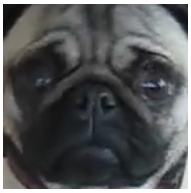
To overcome the weakness of SIFT, many SIFT variants and extensions have also been developed, including principal components analysis based SIFT (PCA-SIFT) (Ke and Sukthankar, 2004), affine invariant extension of SIFT (ASIFT) (Morel and Yu, 2009), gradient location-orientation histogram (GLOH) (Mikolajczyk and Schmid, 2005), and speeded-up robust features (SURF) (Bay *et al.*, 2006).

Like SIFT, PCA-SIFT encodes the salient aspects of the image gradient in the feature points neighbourhood. However, instead of using smoothed Gaussian-weighted orientation histograms, Ke and Sukthankar apply Principal Components Analysis (PCA) to the normalized gradient patch. With the dimensionality reduction, PCA-SIFT gives a simpler and faster local descriptor representation.

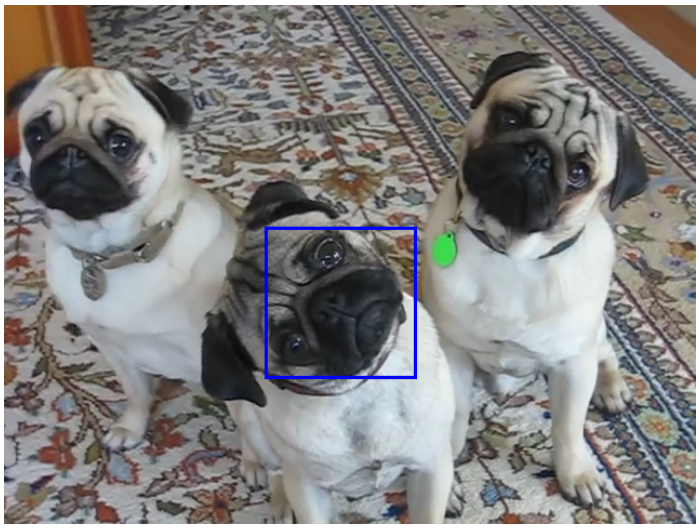
ASIFT extends SIFT to be fully affine invariant. While SIFT normalizes rotations and translations and simulates all scales, ASIFT simulates all distortions caused by a variation of the camera optical axis direction, and then applies the SIFT method. In other words, ASIFT simulates the scale, the camera longitude angle, and the latitude angle and normalizes the translation and rotation, where camera angles are equivalent to image tilt.

These feature-based methods are generally time consuming due to their complicated processes for generating feature descriptors. Especially, in order to make the descriptors invariant to scale change, these methods need to simulate a large scale space which requires intensive computation. Additionally, they may fail to work if the template is relatively small or lightly textured.

To make feature-based techniques more practical, several approaches designed for speeding up the matching process and reducing memory consumption have been proposed. By applying dimensionality reduction techniques, such as PCA (Ke and Sukthankar, 2004)



(a)



(b)

Figure 2.5: Matching examples of BBS.(a) is the given template. (b) shows the matching result in the reference image. Matching area is marked by blue boxes.

and Linear Discriminant Embedding (LDE) (Hua *et al.*, 2007), an original complicated local descriptor could be shortened to accelerate the matching process. It has been shown by Tuytelaars and Schmid (2007), Winder *et al.* (2009), and Calonder *et al.* (2009) that the descriptor vector could be quantized using fewer bits without losing recognition performance. Similarly, hash functions can also help to reduce dimensionality by transforming SIFT descriptors to binary strings whose similarity can be measured by Hamming distance, as shown by Shakhnarovich (2005) and Calonder *et al.* (2010). However, these approaches cannot avoid the complicated processes for generating original complete descriptors.

While SIFT and its variants are mainly designed for general image matching problems, there are also some feature-based algorithms designed specifically for template matching task. A good example is the Best-Buddies Similarity (BBS), which is a parameter-free similarity measure for robust template matching (Dekel *et al.*, 2015; Oron *et al.*, 2016). An example output of BBS can be found in Figure 2.5.

BBS measures the similarity between two sets of points, by counting the number of similar point pairs called Best-Buddies Pairs (BBPs). A pair of points is considered a BBP if each point is the Nearest Neighbour (NN) of the other one in the corresponding point set. The NN is determined by certain distance measures. Suppose that $P = \{p_i\}_{i=1}^{N}$ and $Q = \{q_i\}_{i=1}^{N}$ are two sets of points. By the definition of the BBP, a pair of points $\{p_i, q_j\}$ is BBP when $bb(p_i, q_j, P, Q) = 1$, where

$$bb(p_i, q_j, P, Q) = \begin{cases} 1 & \text{NN}(p_i, Q) = q_j \text{ and NN}(q_j, P) = p_i \\ 0 & \text{otherwise,} \end{cases} \tag{2.12}$$

$\text{NN}(p_i, Q) = \underset{q \in Q}{\text{argmin}}\ d(p_i, q)$, and $d(p_i, q)$ is some distance measures. Then, the BBS can be calculated as the fraction of the number of BBPs out of all the points.

$$\text{BBS}(P, Q) = \frac{1}{N} \cdot \sum_{i=1}^{N} \sum_{j=1}^{N} bb(p_i, q_j, P, Q) \tag{2.13}$$

BBS is robust against moderate geometric deformation and outliers, because it only relies on a relatively small subset of pairs of points and precludes the differences caused by background clutter. Furthermore, it uses rank rather than actual distance value to make it parameter free. In order to make the BBPs more distinctive, BBS takes location information into account when measuring the distance, which is defined as,

$$d(p_i, q_j) = ||p_i^A - q_j^A||_2^2 + \lambda ||p_i^L - q_j^L||_2^2 \tag{2.14}$$

where superscript A denotes the pixel appearance (e.g. RGB values), and superscript L denotes the pixel location, (x, y coordinates within the patch). This measure is not rotation

and scale invariant, making BBS limited in many applications.

For a reference image of M pixels, the computational complexity of BBS is $O(N^2M)$, which is quadratic to the size of the template. Thus, if the input template image is large, the algorithm would be unacceptably slow. It is desirable to expedite the technique.

Our work is also inspired by the scale invariant Centre Surround Extrema detector (CenSurE) by Agrawal *et al.* (2008). CenSurE approximates the Laplacian using binary level centre-surround filters, like octagons filters or hexagons filters, to gain local extrema across all scales. The filters can be computed efficiently with integral images. Schweitzer *et al.* (2013) proposed a template matching technique that combines 45 degrees rotated integral image with non-rotated integral image to improve a corner detector based on Haar wavelets in terms of rotation invariance. It has been shown by the authors that this method can be implemented efficiently using parallel computing architecture like GPU.

Chapter 3

The Proposed Algorithm

In this chapter, we discuss the similarity measurement of the template against each candidate patch using patch features, and introduce our algorithm for rotation and scale invariant template matching pre-screening. In addition, we also take the illumination changes between the template and reference image into consideration and propose corresponding normalizations for defined patch features.

3.1 Patch Features

The proposed algorithm rules out regions with no possible matches based on whether or not the patches in these regions share the same features with the given query template. Patch features employed by conventional template matching techniques are often complex and computationally expensive due to the accuracy and robustness requirements for pinpointing the best match. However, since the proposed pre-processing algorithm only needs to rule out patches that are significantly different from the query template, simple patch features, such as mean, variance and gradient, are sufficient for distinguishing a majority of those

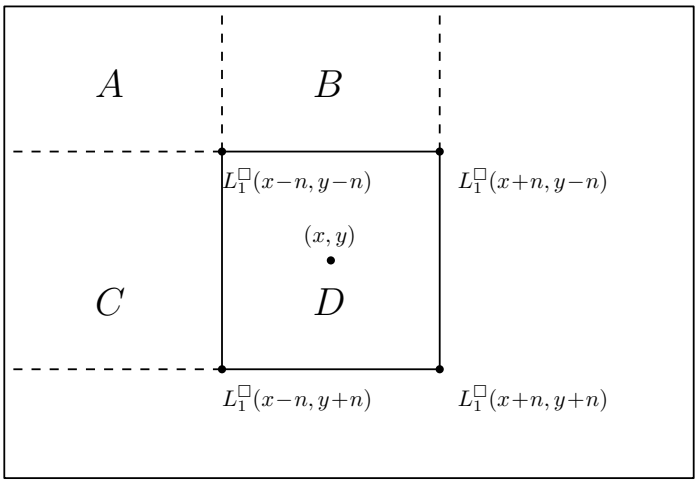


Figure 3.1: The sum $S_1^{\square}(x, y)$ of area D is a simple linear combination of $L_1^{\square}(x+n, y+n)$, $L_1^{\square}(x-n, y+n)$, $L_1^{\square}(x+n, y-n)$, $L_1^{\square}(x-n, y-n)$, which represent the sums of area $A+B+C+D$, $A+C$, $A+B$ and A, respectively

unmatched patches.

The mean of a patch, the inner product of the patch with a box average kernel, can be calculated efficiently by using the inclusion-exclusion principle. As presented in (Crow, 1984), it only needs one addition and two subtractions to calculate the sum of the pixel intensity of an arbitrary rectangular area using a summed area table, which is also known as integral image. The idea is that, if we know the sum $L_1^{\square}(x, y)$ of a rectangular area with top-left corner $(1, 1)$ and bottom-right corner (x, y) for any given pixel (x, y) of image I, as shown in Figure 3.1, then we can calculate the sum $S_1^{\square}(x, y)$ of any $2n \times 2n$ patch centred at (x, y) as follows,

$$\begin{aligned} S_1^{\square}(x, y) = & \sum_{i=y-n+1}^{y+n} \sum_{j=x-n+1}^{x+n} I(i, j) \\ = & L_1^{\square}(x+n, y+n) - L_1^{\square}(x-n, y+n) \\ & - L_1^{\square}(x+n, y-n) + L_1^{\square}(x-n, y-n), \end{aligned} \tag{3.1}$$

-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3
-3	-2	-1	0	1	2	3

(a) Horizontal

-3	-3	-3	-3	-3	-3	-3
-2	-2	-2	-2	-2	-2	-2
-1	-1	-1	-1	-1	-1	-1
0	0	0	0	0	0	0
1	1	1	1	1	1	1
2	2	2	2	2	2	2
3	3	3	3	3	3	3

(b) Vertical

Figure 3.2: The extended 7×7 Prewitt kernels, which are basically linear gradient ramps.

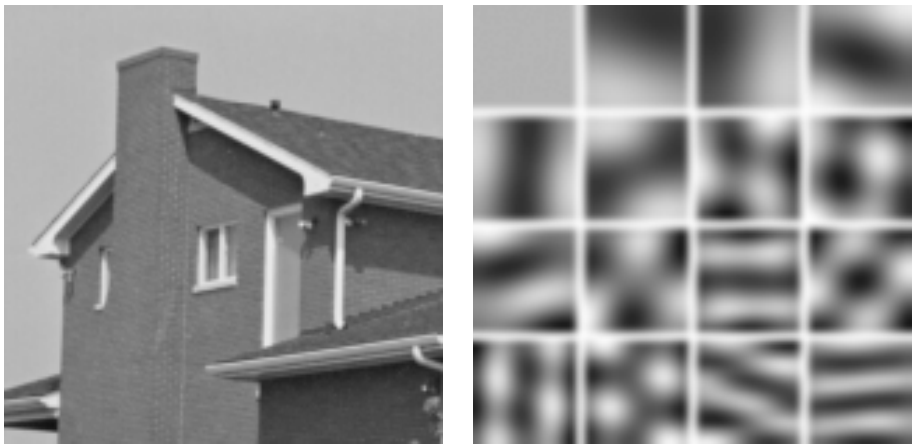
where the summed area table $L_1^{\square}(x, y)$, by its definition, is,

$$L_1^{\square}(x, y) = \sum_{i=1}^{y} \sum_{j=1}^{x} I(i, j). \tag{3.2}$$

Given the sum of a patch centred at (x, y), the mean $S_{\mu}^{\square}(x, y)$ of the patch is,

$$S_{\mu}^{\square}(x, y) = S_1^{\square}(x, y)/(2n)^2. \tag{3.3}$$

The summed area table $L_1^{\square}(x, y)$ for image I can be constructed efficiently by using a pass of horizontal cumulative sum on every row of I followed by a pass of vertical cumulative sum on every column. Since the cumulative sums of different rows (or columns) are independent, it is easy to accelerate the construction process with parallel computing architecture, such as GPU, by calculating the cumulative sums of rows (or columns) concurrently. After the linear-time construction of the summer area table $L_1^{\square}(x, y)$, each of the following query for the intensity sum $S_1^{\square}(x, y)$ or mean $S_{\mu}^{\square}(x, y)$ of an arbitrary patch only requires constant time regardless of the size the patch.



(a) Input image (b) First 16 components

Figure 3.3: A sample image and the first 16 principle components of all the 16×16 patches in the image Deledalle *et al.* (2011).

In addition to the mean of a patch, the proposed algorithm also employs two other linear features, the horizontal and vertical gradients. The horizontal and vertical gradient of a patch, representing the rate of pixel intensity change of the patch from left to right and from top to bottom respectively, are the inner products of the patch with the extended Prewitt kernels (Kekre and Gharge, 2010) of the same size. The extended Prewitt kernels are basically linear gradient ramps as shown in Figure 3.2, and they are defined for an $m \times m$ patch as follows,

$$\begin{aligned} P_x(x, y) &= x - \frac{m+1}{2}, \\ P_y(x, y) &= y - \frac{m+1}{2}, \end{aligned} \tag{3.4}$$

where the indexes x and y start from 1.

Choosing horizontal and vertical gradients as patch features is based on the fact that the first three principle components of natural image patches in Principle Component Analysis

(PCA) are the box average kernel, vertical extended Prewitt kernel and horizontal extended Prewitt kernel, respectively (as shown in Figure 3.3). Thus, for distinguishing patches, these kernels have the strongest discriminating power among all the combinations of any three linear features.

In a similar fashion as the sum of a patch as in Eq. 3.1, gradient of a patch can also be calculated efficiently by using the inclusion-exclusion principle. For instance, the horizontal gradient $S_x^{\square}(x, y)$ of a $2n \times 2n$ patch centred at (x, y) is,

$$\begin{aligned} S_x^{\square}(x,y) = & \sum_{i=y-n+1}^{y+n} \sum_{j=x-n+1}^{x+n} (j-x) \cdot I(i,j) \\ = & L_x^{\square}(x+n, y+n) - L_x^{\square}(x-n, y+n) \\ & - L_x^{\square}(x+n, y-n) + L_x^{\square}(x-n, y-n) \\ & - x \cdot S_1^{\square}(x,y), \end{aligned} \tag{3.5}$$

where $L_x^{\square}(x, y)$, the summed area map of image $j \cdot I(i, j)$, is,

$$L_x^{\square}(x,y) = \sum_{i=1}^{y} \sum_{j=1}^{x} j \cdot I(i,j). \tag{3.6}$$

Similarly, the vertical gradient $S_y^{\square}(x,y)$ of the patch is tractable with the same technique.

$$\begin{aligned} S_y^{\square}(x,y) = &\sum_{i=y-n+1}^{y+n} \sum_{j=x-n}^{x+n+1} (i-y)\cdot I(i,j) \\ =&L_y^{\square}(x+n,y+n) - L_y^{\square}(x-n,y+n) \\ &- L_y^{\square}(x+n,y-n) + L_y^{\square}(x-n,y-n) \\ &- y\cdot S_1^{\square}(x,y), \end{aligned} \tag{3.7}$$

where the summed area table $L_y^{\square}(x,y)$ is defined as,

$$L_y^{\square}(x,y) = \sum_{i=1}^{y}\sum_{j=1}^{x} i\cdot I(i,j). \tag{3.8}$$

Additionally, in order to make the patch gradient independent to the patch size, we also introduce two normalization terms as follows,

$$S_x^{\square}(x,y) = \frac{\sum\limits_{i=y-n+1}^{y+n}\sum\limits_{j=x-n+1}^{x+n}(j-x)\cdot I(i,j)}{\sum\limits_{i=y-n+1}^{y+n}\sum\limits_{j=x-n+1}^{x+n}|j-x|}, \tag{3.9}$$

$$S_y^{\square}(x,y) = \frac{\sum\limits_{i=y-n+1}^{y+n}\sum\limits_{j=x-n+1}^{x+n}(i-y)\cdot I(i,j)}{\sum\limits_{i=y-n+1}^{y+n}\sum\limits_{j=x-n+1}^{x+n}|i-y|}, \tag{3.10}$$

which are the sum of the absolute values of the employed extended Prewitt kernels.

Another patch feature employed by the proposed algorithm is variance. Given the mean

$S_{\mu}^{\square}(x,y)$ and second moment $S_{2}^{\square}(x,y)$ of a $2n \times 2n$ patch, the variance $S_{\sigma}^{\square}(x,y)$ of the patch can be obtained as,

$$S_{\sigma}^{\square}(x,y) = S_{2}^{\square}(x,y)/(2n)^2 - [S_{\mu}^{\square}(x,y)]^2. \tag{3.11}$$

Since the second moment $S_{2}^{\square}(x,y)$ of each patch,

$$S_{2}^{\square}(x,y) = \sum_{i=x-n+1}^{x+n} \sum_{j=y-n+1}^{y+n} [I(i,j)]^2, \tag{3.12}$$

only requires constant time to calculate with the summed area table $L_{2}^{\square}(x,y)$, we can acquire the variance of an arbitrary patch efficiently as well.

Specifically, the summed area table $L_{2}^{\square}(x,y)$, and the second moment $S_{2}^{\square}(x,y)$ would be carried out like follows,

$$\begin{aligned} S_{2}^{\square}(x,y) = & \sum_{i=y-n+1}^{y+n} \sum_{j=x-n+1}^{x+n} [I(i,j)]^2 \\ = & L_{2}^{\square}(x+n, y+n) - L_{2}^{\square}(x-n, y+n) \\ & - L_{2}^{\square}(x+n, y-n) + L_{2}^{\square}(x-n, y-n), \end{aligned} \tag{3.13}$$

where,

$$L_{2}^{\square}(x,y) = \sum_{i=1}^{y} \sum_{j=1}^{x} [I(i,j)]^2. \tag{3.14}$$

3.2 Rotation Invariance

The patch features discussed in the previous section are excellent indicators for ruling out unmatched patches if the object in the query template and reference image shares the same orientation and scale. However, if that is not the case, a small mismatch between the orientations of the query template and reference image could result in distinct patch features for the same object, causing mistakes in matching the object. This over-sensitivity to orientation is due to two problems: first, some of patch features like horizontal and vertical gradients are not rotation invariant; second, the shape of the template is also not rotation invariant.

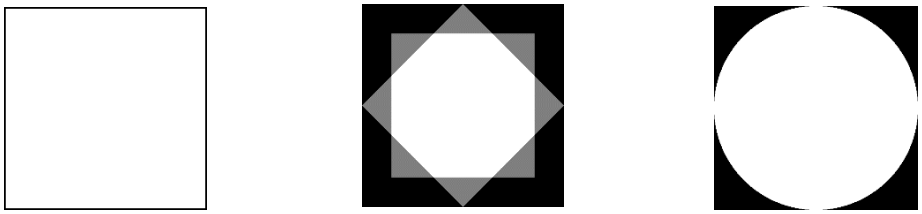
The first problem can be easily resolved by replacing the horizontal and vertical gradients with the magnitude of gradient,

$$S_m^{\square}(x,y) = \sqrt{[S_x^{\square}(x,y)]^2 + [S_y^{\square}(x,y)]^2}, \tag{3.15}$$

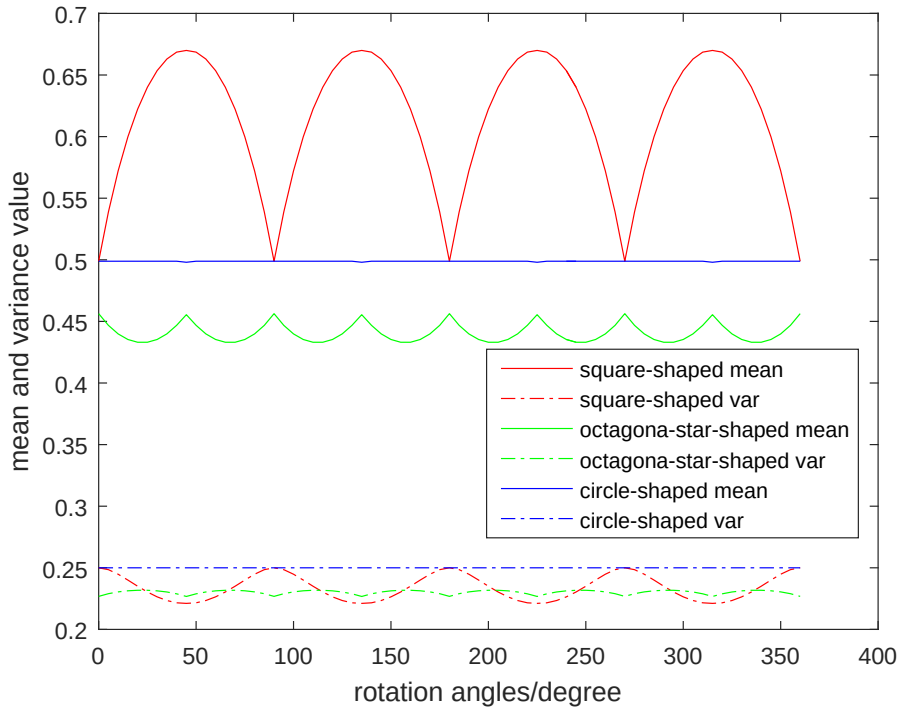
which is robust against rotation and still simple to calculate.

To solve the second problem, we need to change the shape of the template. Commonly, the query template is a square image patch provided by the user. Given the template, the pre-processing algorithm then examines the features of every square patch of the same size as the template in the reference image. The drawback of using square patch is the area covered by a square patch are variant to image rotation. For instance, the reference image shown in Figure 3.4 is a black framed box on white background. A rotation of the reference image moves some of the white background pixels into the window centred at the box while cropping out some of the black frame pixels. The resulting change in the set of pixels in the window alters all its patch features such as the mean and variance as in Figure 3.4.

(a) Template (b) Reference image



(c) Square (d) Octagonal star (e) Circle



(f) The mean and variance of a patch plotted as a function of image rotation angle.

Figure 3.4: Octagonal-star-shaped template is much more robust against rotation than square template.

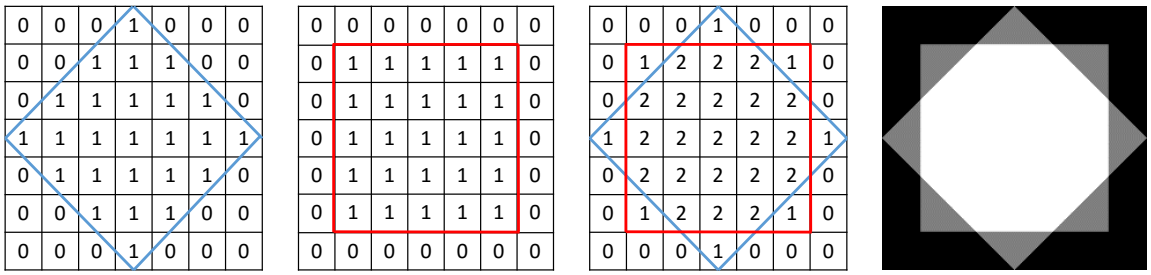


Figure 3.5: An example of octagonal-star-shaped 7×7 template (the third figure), which is the superposition of a square (the second figure) and a diamond (the first figure).

Ideally, the best template shape for rotation invariant matching is a circle, but it is not possible to accelerate the feature calculation of an arbitrary circular patch using the inclusion-exclusion technique introduced previously. Thus, we propose octagonal-star-shaped template, which approximates to a circle but still has efficient feature calculation algorithm. As shown in Figure 3.4, the mean and variance value of octagonal-star-shaped template fluctuate within a narrow bound, and hence are much more stable and robust against rotation than square-shaped template.

An octagonal star is the superposition of a square and a diamond (square rotated by $45°$) of the same size, as depicted in Figure 3.5. The mean $S_\mu(x, y)$ of an octagonal-star-shaped template, for example, is the arithmetic average of the mean $S_\mu^{\square}(x, y)$ of the square and the mean $S_\mu^{\diamond}(x, y)$ of the diamond in the template, i.e.,

$$S_\mu(x, y) = \frac{1}{2}[S_\mu^{\square}(x, y) + S_\mu^{\diamond}(x, y)], \tag{3.16}$$

Similar to the previous rectangular case, if the sum $L_1^{\diamond}(x, y)$ of the triangular area with

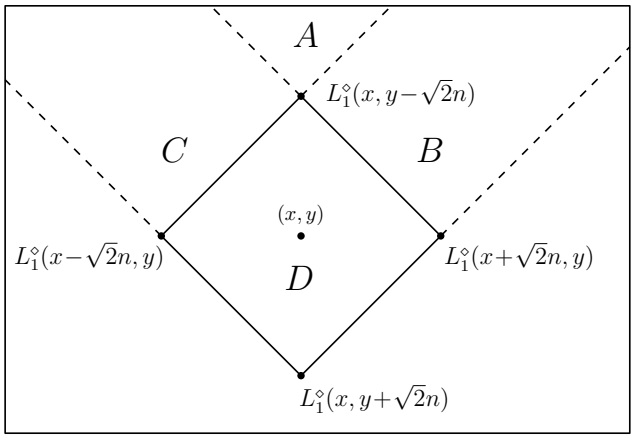


Figure 3.6: The sum $S_1^{\diamond}(x, y)$ of area D is a simple linear combination of $L_1^{\diamond}(x, y+\sqrt{2}n), L_1^{\diamond}(x-\sqrt{2}n, y), L_1^{\diamond}(x+\sqrt{2}n, y), L_1^{\diamond}(x, y-\sqrt{2}n)$, which represent the sums of area $A+B+C+D$, $A+C$, $A+B$ and A, respectively

right angle vertex (x, y), as shown in Figure 3.6, is known for any x, y, where $L_1^{\diamond}(x, y)$ is,

$$L_1^{\diamond}(x, y) = \sum_{i=1}^{y-|x-j|} \sum_{j=\max(x-y,1)}^{\min(x+y,M)} I(i, j), \tag{3.17}$$

and M is the width of the image. Then, by the inclusion-exclusion principle, we can calculate the sum $S_1^{\diamond}(x, y)$ and mean $S_{\mu}^{\diamond}(x, y)$ of a $2n \times 2n$ diamond area centred at (x, y) using one addition and two subtractions as follows (Lienhart and Maydt, 2002; Messom and Barczak, 2006),

$$\begin{aligned} S_1^{\diamond}(x, y) =& L_1^{\diamond}(x, y+\sqrt{2}n) - L_1^{\diamond}(x-\sqrt{2}n, y) \\ & - L_1^{\diamond}(x+\sqrt{2}n, y) + L_1^{\diamond}(x, y-\sqrt{2}n), \end{aligned} \tag{3.18}$$

$$S_{\mu}^{\diamond}(x, y) = S_1^{\diamond}(x, y)/(2n)^2. \tag{3.19}$$

Same as $L_1^{\square}(x,y)$, the construction of the summed triangular area map $L_1^{\diamond}(x,y)$ is tractable in linear time using two diagonal passes of cumulative sum. Besides the mean, other patch features, such as the variance and magnitude of gradient, of a octagonal-star-shaped template can also be computed efficiently by combining the features of the square and diamond areas in the template. The corresponding summed area tables for diamond areas are listed as follows,

$$L_2^{\diamond}(x,y) = \sum_{i=1}^{y-|x-j|} \sum_{j=\max(x-y,1)}^{\min(x+y,M)} [I(i,j)]^2, \tag{3.20}$$

$$L_x^{\diamond}(x,y) = \sum_{i=1}^{y-|x-j|} \sum_{j=\max(x-y,1)}^{\min(x+y,M)} j \cdot I(i,j), \tag{3.21}$$

$$L_y^{\diamond}(x,y) = \sum_{i=1}^{y-|x-j|} \sum_{j=\max(x-y,1)}^{\min(x+y,M)} i \cdot I(i,j). \tag{3.22}$$

Besides the above mentioned rotation invariant features, we could also calculate the orientation of a patch gradient based on established summed area tables as follows,

$$S_\theta(x,y) = \mathrm{atan2}(S_y(x,y), S_x(x,y)), \tag{3.23}$$

where $S_y(x,y)$ and $S_x(x,y)$ are the addition of the gradient of the square area and the diamond area in vertical and horizontal directions, respectively. Although the orientation of gradient is not used as a patch feature in the pre-processing stage, the conventional accurate template matching technique in second stage can utilize this orientation information to align a patch with the query template for robust matching against image rotation.

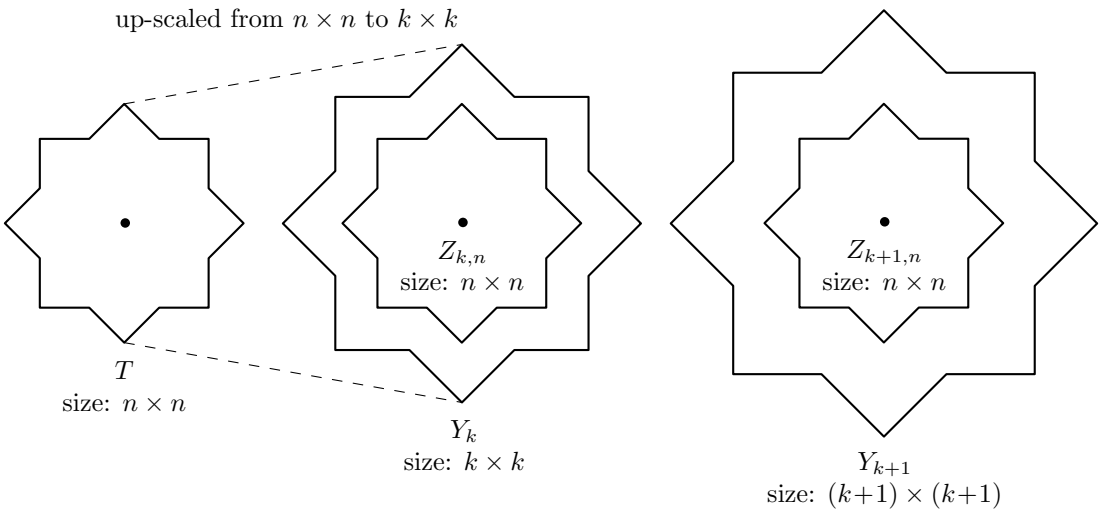


Figure 3.7: Templates in different scales.

3.3 Scale Invariance

Another challenge for the proposed pre-processing algorithm is how to handle the scale difference between the query template and reference image. In the previous sections, we assume that the scales of the query template and reference image are identical, hence the best matched patch must be of the exact same size as the query template. However, this assumption is impractical in most real-world applications; the scale information is usually unknown and the best matched patch may be larger or smaller than the query template.

One solution to this scale invariant problem is simply to scale the query octagonal-star shaped template to difference sizes and then check if some of the resized templates have a well matched patch of the same size in the reference image (Hinterstoisser *et al.*, 2012). If the best matched patch can be any size from $\alpha n \times \alpha n$ to $\beta n \times \beta n$, where n is the side length of the query template and α, β are some constants, then there are $\beta n - \alpha n = O(n)$ different possible template scales to examine. In the case of our pre-processing algorithm, as comparing the features of a template to all patches of the same size requires $O(M^2 +$

$n^2)$ time for an $M \times M$ reference image, the overall time complexity for scale invariant screening with the aforementioned method is $O(M^2n + n^3)$.

The exhaustive search strategy is inefficient and unsuited for our goal of fast screening of unmatched regions, especially when the query template is large. But this method can be greatly accelerated if we only check patches of a few different sizes in the reference image instead of enumerating every possible sizes. The idea is based on an assumption that, if two patches of the same size have matched central area, then these two patches might also match. For instance, given an $n \times n$ query template T, suppose that image Y_k of size $k \times k$ is an up-scaled version of T as shown in Figure 3.7, and the patch feature vector $f_{k,n}$ of the $n \times n$ centre of Y_k, namely $Z_{k,n}$, matches a patch P_n in the reference image, then the $k \times k$ patch P_k centred at the same location as P_n might also match the features of Y_k. Similarly, given Y_{k+1}, a $(k+1) \times (k+1)$ version of T, if it matches a patch P_{k+1} of the same size, their $n \times n$ central areas, $Z_{k+1,n}$ and P_n, should have matching features. Therefore, to find the matched patches of either size $k \times k$ or $(k+1) \times (k+1)$, we only have to look through all the $n \times n$ patches in the reference image; there is no need for calculating the features for patches of different sizes.

The assumption that matched central areas indicate matched patches is not always true obviously, however, it only slightly increases the possibility of mistaking a wrong patch as matched, as long as the examined centre area is sufficiently large, i.e., n is not much smaller than k. Moreover, the best match should have a matched central area to the query template hence not affected by the assumption. Therefore, despite the limitation of the assumption, the pre-processor can still rule out a great amount of unlikely matched areas without removing the best match for the next stage template matching algorithm.

Although this method can process the templates of several different scales with only the

patches of one fixed size, it still needs to compare the features of a patch with the features of the query template of every scale. Thus, the time complexity for comparing all the templates with all the patches is still $O(M^2n)$. To make the feature comparison function more efficient, we can aggregate the features of the templates of different scales together into a feature set F_n,

$$F_n = \{f_{k,n} \,|\, n \leq k \leq \lambda n\}, \tag{3.24}$$

which includes all the feature vectors of the scaled templates of size from $n \times n$ to $\lambda n \times \lambda n$. Now given F_n, we can examine if an $n \times n$ patch P matches $Z_{k,n}$ for some $k \in [n, \lambda n]$ by testing the membership of the feature vector of P in set F_n. If the size of the best matched patch ranges from $\alpha n \times \alpha n$ to $\beta n \times \beta n$, then we need to repeat the process $\log_\lambda(\beta/\alpha)$ times for different scale ranges.

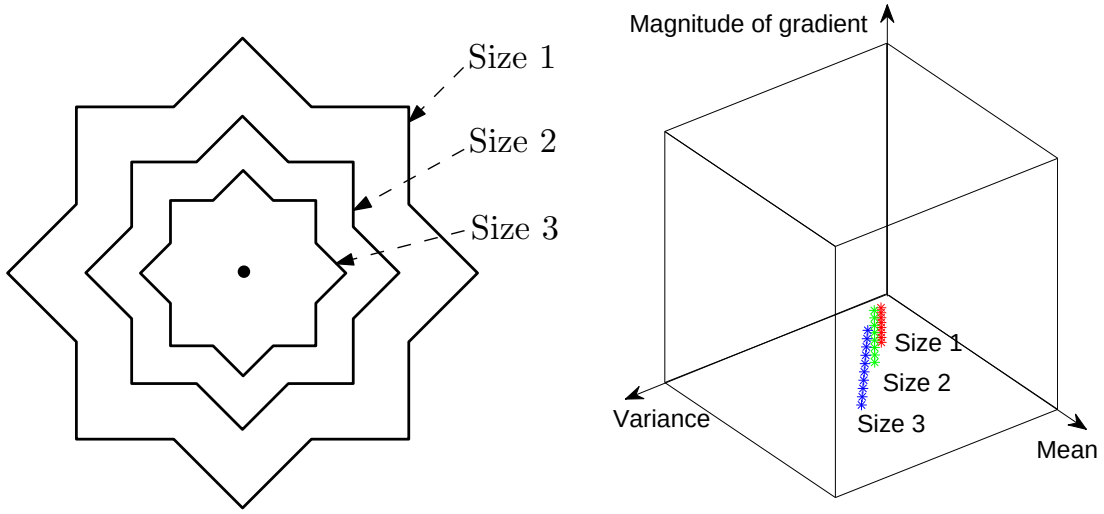
The feature set F_n can be implemented using a membership array $\hat{F}_n$ with $\hat{F}_n(Q(f)) = 1$ for any $f \in F_n$, where $Q(f)$ is a quantizer for the feature vector. For simplicity's sake, assume that there is only one feature to consider and $Q(f) = \lfloor f/q + \frac{1}{2} \rfloor$ is a uniform quantizer with quantization factor q. For some $f \in F_n$, if both $Q(f - q/2)$ and $Q(f + q/2)$ are marked as 1 in $\hat{F}_n$ in addition to $Q(f)$, then, given a feature g, $\hat{F}_n(Q(g)) = 1$ as long as $|f - g| < q/2$. Thus, quantization factor q is a parameter that sets how similar the features of two patches should be before counting them as matched.

This efficient scale invariant pre-processing algorithm is summarized in Algorithm 1. Since a membership array only requires constant time to initialize and constant time to access one element regardless of the size of the array (Storer, 2001), the time complexity for constructing F_n is $O(n^3)$ and the time complexity for testing the features of all the patches is $O(M^2)$. Overall, the proposed scale invariant pre-processing algorithm requires

Algorithm 1: Screening pre-processor for scale invariant template matching

Data: I, reference image
Data: T, query template
Data: $[\alpha, \beta]$, scale range
Result: R, set of possible matched patches

begin
  $\lambda \leftarrow \sqrt{2}$
  $n \leftarrow \text{get\_width}(T)$
  $R \leftarrow \varnothing$
  $m \leftarrow \beta n$
  while $m \geq \alpha n$ **do**
    $m \leftarrow m/\lambda$
    $F_m \leftarrow \varnothing$
    for $k \leftarrow m$ **to** $m\lambda$ **do**
      $Y_k \leftarrow \text{scale}(T, k \times k)$
      $Z_{k,m} \leftarrow \text{crop\_center}(Y_k, m \times m)$
      $f_{k,m} \leftarrow Q(\text{features}(Z_{k,m}))$
      $F_m \leftarrow F_m \cup \{f_{k,m}\}$
    foreach $m \times m$ *patch* P *in* I **do**
      if $Q(\text{features}(P)) \in F_m$ **then**
        $R = R \cup \{P\}$



(a) Central areas of different sizes

(b) Vector set F_n for central areas of different sizes

Figure 3.8: Features from different central areas of a patch are utilized together to improve matching accuracy.

$O(M^2 + n^3)$ time. Considering that the reference image is generally much larger than the query template, the increase in the asymptotic computational complexity from $O(M^2+n^2)$ due to the addition of scale invariant support is insignificant. Additionally, one important observation in our implementation is that, the pixel-wise increase in the template's scale is not necessary while constructing feature set F_n, because the proposed statistical patch features are relatively stable with slight scale difference. Therefore, a proportional growth from size m to size $m\lambda$ is accepted in our algorithm, and proves to achieve the same performance as exhaustive increasing in scale size. In this sense, the asymptotic computational complexity of the proposed technique is still $O(M^2 + n^2)$.

Patch features of different scales calculated for scale invariant matching benefit the accuracy of matching as well. As discussed previously, we consider that a $k \times k$ patch P_k matches template Y_k of size $k \times k$ if their $n \times n$ central areas P_n and $Z_{k,n}$ share the similar features. In addition to P_n and $Z_{k,n}$, if the smaller $(n/\lambda) \times (n/\lambda)$ central areas $P_{n/\lambda}$ and

$Z_{k,n/\lambda}$ also have matched features, patch P_k is even more likely to be a match to the query template. Therefore, given a query template, the feature comparison process should not be limited only to the large central area; the additional features from smaller centre areas, as illustrated in Figure 3.8a, can be used to provide extra details about a patch, making the matching process more accurate and reliable. Figure 3.8b plots scale invariant feature vector set F_n for each central area of a query template. In our scale invariant algorithm, if any one of three central areas of a patch does not match the features of the corresponding areas of the template, the patch is marked as unmatched. Since these features of different scales $(n, n/\lambda, n/\lambda^2, \ldots)$ have already been calculated for scale invariant matching, the extra cost for utilizing these features is negligible.

3.4 Illumination Invariance

Different illumination conditions may form different images of an object. Thus, in real-world template matching applications, the illumination difference between the given template and reference image is another common problem causing mismatches. In our algorithm, all the employed patch features, i.e., mean, variance and the magnitude of gradient, are also inevitably affected by the illumination differences. One way to make our proposed algorithm robust against varying illumination conditions, is to normalize existing patch features.

The formation of images of an object under different lighting conditions is commonly modelled as follows,

$$I(i,j) = R(i,j) \cdot B(i,j), \tag{3.25}$$

where $I(i,j)$ is the image pixel value, $R(i,j)$ is the reflectance and $B(i,j)$ is the intensity of illumination at each point (i,j). Here, the reflectance $R(i,j)$ is determined by the characteristics of the object's surface, while the illumination $B(i,j)$ is a function of the brightness of lighting source. Since $R(i,j)$ is constant to lighting conditions for an object, the change of pixel value $I(i,j)$ is solely a result of the intensity change of illumination $B(i,j)$.

Assume that a pixel value $T(i_1,j_1)$ in the given template is,

$$T(i_1,j_1) = R(i_1,j_1) \cdot B_T(i_1,j_1), \tag{3.26}$$

and the value $I(i_2,j_2)$ of the corresponding pixel in the best matched patch from the reference image is,

$$I(i_2,j_2) = R(i_2,j_2) \cdot B_I(i_2,j_2). \tag{3.27}$$

If every other condition stays the same in the query template and reference image, the ratio of the pixel value $T(i_1,j_1)$ to $I(i_2,j_2)$ should be the same as the ratio of illumination level $B_T(i_1,j_1)$ to $B_I(i_2,j_2)$, i.e.,

$$\frac{I(i_2,j_2)}{T(i_1,j_1)} = \frac{B_I(i_2,j_2)}{B_T(i_1,j_1)} = \alpha, \tag{3.28}$$

where α represents the ratio of the two illumination levels.

By Eq. 3.28, we can rewrite the proposed patch features of a $2n \times 2n$ square area centred

at position (x,y) as follows,

$$\begin{aligned} S_{\mu}^{\square}(x,y) =& \frac{\sum_{i_2=y-n+1}^{y+n}\sum_{j_2=x-n+1}^{x+n} I(i_2,j_2)}{(2n)^2} \\ =& \frac{\sum_{i_1=1}^{2n}\sum_{j_1=1}^{2n} \alpha \cdot T(i_1,j_1)}{(2n)^2} \\ =& \alpha \cdot T_{\mu}^{\square}, \end{aligned} \tag{3.29}$$

$$\begin{aligned} S_{\sigma}^{\square}(x,y) =& S_2^{\square}(x,y)/(2n)^2 - [S_{\mu}^{\square}(x,y)]^2 \\ =& \frac{\sum_{i_2=y-n+1}^{y+n}\sum_{j_2=x-n+1}^{x+n} [I(i_2,j_2)]^2}{(2n)^2} - [S_{\mu}^{\square}(x,y)]^2 \\ =& \frac{\sum_{i_1=1}^{2n}\sum_{j_1=1}^{2n} \alpha^2 \cdot [T(i_1,j_1)]^2}{(2n)^2} - \alpha^2 \cdot [T_{\mu}^{\square}]^2 \\ =& \alpha^2 \cdot T_{\sigma}^{\square}, \end{aligned} \tag{3.30}$$

$$\begin{aligned} S_{m}^{\square}(x,y) =& \sqrt{[S_x^{\square}(x,y)]^2 + [S_y^{\square}(x,y)]^2} \\ =& \sqrt{\left[\frac{\sum_{i_2=y-n+1}^{y+n}\sum_{j_2=x-n+1}^{x+n}(j_2-x)\cdot I(i_2,j_2)}{\sum_{i_2=y-n+1}^{y+n}\sum_{j_2=x-n+1}^{x+n}|j_2-x|}\right]^2 + \left[\frac{\sum_{i_2=y-n+1}^{y+n}\sum_{j_2=x-n+1}^{x+n}(i_2-y)\cdot I(i_2,j_2)}{\sum_{i_2=y-n+1}^{y+n}\sum_{j_2=x-n+1}^{x+n}|i_2-y|}\right]^2} \\ =& \sqrt{\left[\frac{\sum_{i_1=1}^{2n}\sum_{j_1=1}^{2n}(j_1-\frac{2n+1}{2})\cdot \alpha \cdot T(i_1,j_1)}{\sum_{i_1=1}^{2n}\sum_{j_1=1}^{2n}|j_1-\frac{2n+1}{2}|}\right]^2 + \left[\frac{\sum_{i_1=1}^{2n}\sum_{j_1=1}^{2n}(i_1-\frac{2n+1}{2})\cdot \alpha \cdot T(i_1,j_1)}{\sum_{i_1=1}^{2n}\sum_{j_1=1}^{2n}|i_1-\frac{2n+1}{2}|}\right]^2} \\ =& \alpha \cdot T_{m}^{\square}. \end{aligned} \tag{3.31}$$

Using the same method, we can also reformulate the mean, variance and the magnitude of

gradient of the diamond area in our octagonal-star-shaped template.

Now, to make the variance feature of a patch invariant to illumination changes, we can normalize the variance with the square of the patch mean as follows,

$$\widehat{S}_{\sigma}^{\square}(x,y) = \frac{S_{\sigma}^{\square}(x,y)}{[S_{\mu}^{\square}(x,y)]^2} = \frac{\alpha^2 \cdot T_{\sigma}^{\square}}{\alpha^2 \cdot [T_{\mu}^{\square}]^2} = \frac{T_{\sigma}^{\square}}{[T_{\mu}^{\square}]^2}, \tag{3.32}$$

where the $\widehat{S}_{\sigma}^{\square}(x,y)$ represents the normalized patch variance. Similarly, the magnitude of patch gradient can be normalized with the patch mean as,

$$\widehat{S}_{m}^{\square}(x,y) = \frac{S_{m}^{\square}(x,y)}{S_{\mu}^{\square}(x,y)} = \frac{\alpha \cdot T_{m}^{\square}}{\alpha \cdot T_{\mu}^{\square}} = \frac{T_{m}^{\square}}{T_{\mu}^{\square}}, \tag{3.33}$$

where the $\widehat{S}_{m}^{\square}(x,y)$ represents the normalized magnitude of patch gradient.

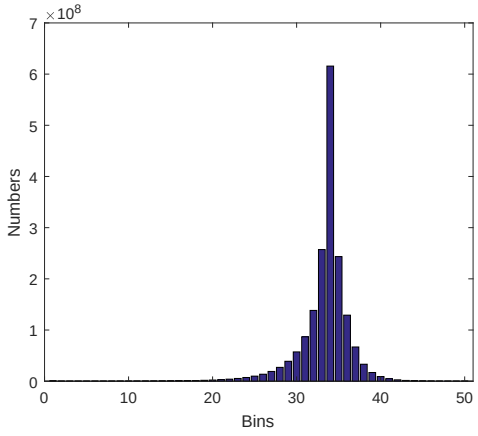
Taking our scale invariance algorithm into account, patch features calculated for different scales can be utilized with different combinations for more accurate matching. Specifically, for a certain scale, the three concentric central areas of different sizes, as shown in Figure 3.8a, can be computed and compared with their counterparts in the template, respectively. Thus, based on this, the ratio between the three different patch means, which are shown as follows, is also a feature invariant to illumination changes,

$$\frac{S_{\mu_2}^{\square}(x,y)}{S_{\mu_1}^{\square}(x,y)} = \frac{\alpha \cdot T_{\mu_2}^{\square}}{\alpha \cdot T_{\mu_1}^{\square}} = \frac{T_{\mu_2}^{\square}}{T_{\mu_1}^{\square}}, \tag{3.34}$$

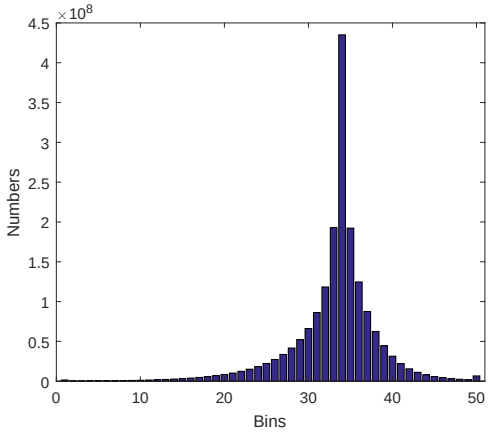
$$\frac{S_{\mu_3}^{\square}(x,y)}{S_{\mu_1}^{\square}(x,y)} = \frac{\alpha \cdot T_{\mu_3}^{\square}}{\alpha \cdot T_{\mu_1}^{\square}} = \frac{T_{\mu_3}^{\square}}{T_{\mu_1}^{\square}}, \tag{3.35}$$

where $S_{\mu_i}^{\square}(x,y)$ and $T_{\mu_i}^{\square}$ represent the mean of central area of size $i, (i \in \{1,2,3\})$ in any patch and the given template respectively.

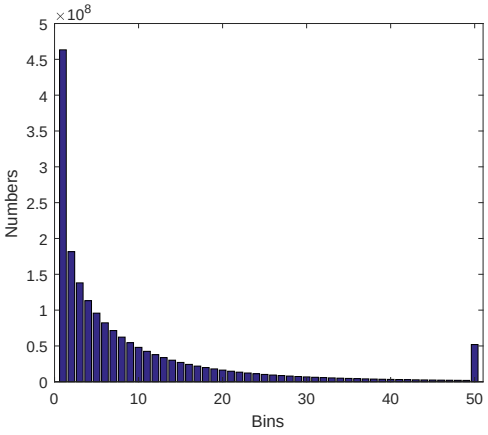
Based on the above discussion, the ratio of patch means, normalized patch variance and



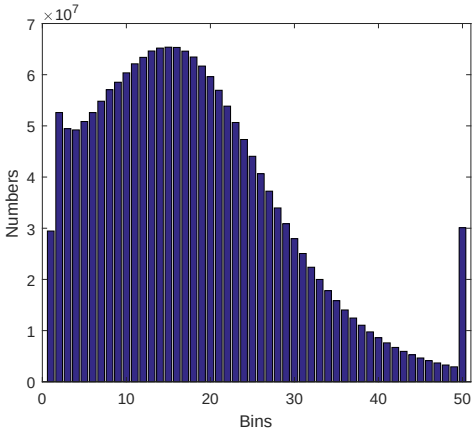
(a) The ratio of $S^{\square}_{\mu_2}(x,y)$ and $S^{\square}_{\mu_1}(x,y)$.



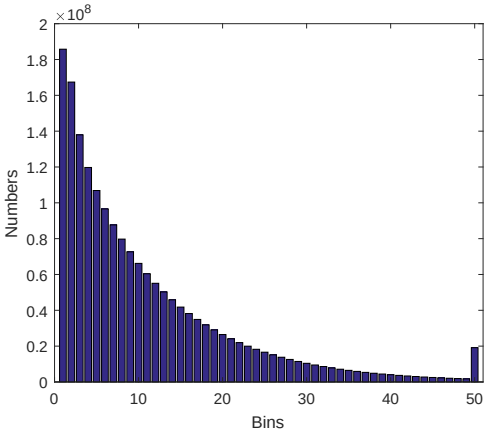
(b) The ratio of $S^{\square}_{\mu_3}(x,y)$ and $S^{\square}_{\mu_1}(x,y)$.



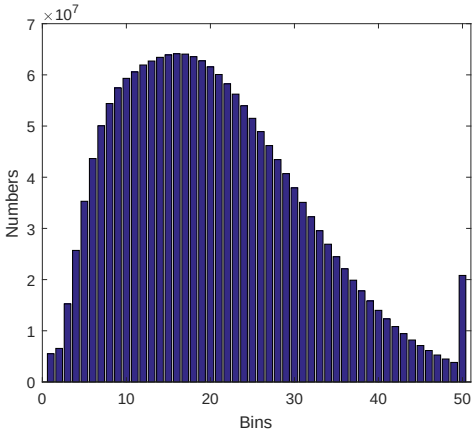
(c) $\widehat{S}^{\square}_{\sigma}(x,y)$.



(d) $[\widehat{S}^{\square}_{\sigma}(x,y)]^{0.4}$



(e) $\widehat{S}^{\square}_{m}(x,y)$.



(f) $[\widehat{S}^{\square}_{m}(x,y)]^{0.5}$

Figure 3.9: Distributions of illumination invariant patch features.

normalized magnitude of patch gradient are selected as our illumination invariant patch features. To find the optimal quantization interval in the previously defined membership array, we analyse the distribution of these patch features. After obtaining patch features of all 63×63 patches in over 10,000 images in COCO data set (Lin *et al.*, 2014), the distributions of the three features have been shown in Figure 3.9, where all the results are uniformly quantized into 50 bins. The distributions of $\widehat{S}_{\sigma}^{\square}(x, y)$ and $\widehat{S}_{m}^{\square}(x, y)$ are concentrated on small values, which are not ideal for uniform quantization. By applying a power function to $\widehat{S}_{\sigma}^{\square}(x, y)$ and $\widehat{S}_{m}^{\square}(x, y)$, both of the two distributions can be reshaped more evenly. In our implementation, the powers for $\widehat{S}_{\sigma}^{\square}(x, y)$ and $\widehat{S}_{m}^{\square}(x, y)$ are set empirically as 0.4 and 0.5, respectively, as shown in Figure 3.9d and Figure 3.9f.

Chapter 4

The Parallel Implementation

Computational efficiency is vital to the proposed screening algorithm. As discussed in the previous section, by using the summed area tables and the inclusion-exclusion principle, the computational complexity of the proposed algorithm is only $O(M^2 + n^2)$ for pre-processing an $M \times M$ reference image with an $n \times n$ template. As in general, M is much larger than n, the asymptotic time complexity of the algorithm is dominated by the M^2 term. In practice, the majority of the total processing time of our algorithm implementation is indeed spent on establishing summed area tables and searching through all candidate patches in the reference image, while in comparison, the time cost for building feature sets for the template is negligible. As mentioned in Section 3.1 and Section 3.2, it is possible to calculate the summed area table in parallel, and additionally, the feature testing for each candidate patch in the reference image is mutually independent hence can be carried out concurrently. Thus, we can further accelerate our algorithm with the help of modern parallel computing architecture like GPU. In this Chapter, we present a highly efficient GPU-based parallel version of the proposed pre-screening algorithm and compare the performances of the CPU-based and GPU-based implementations.

4.1 Fundamentals of CUDA

The computer unified device architecture (CUDA) platform developed by NVIDIA is a parallel computing architecture that enables the GPU to solve generic computational problems. The advantage of CUDA is that it greatly simplifies the GPU-based parallel programming and can be deployed to many modern computing devices such as personal computers, smart phones, etc. In this section, we introduce some of the basic concepts and terminologies of GPU and CUDA technologies to help the reader grasp the idea of our GPU-based screening algorithm.

The superior computational power of GPU comes from its single instruction multiple data (SIMD) architecture, where an array of data can be processed simultaneously using a single instruction. The SIMD architecture is very efficient in applications that involve a large set of data but without many complicated flow controls. A GPU is built around an array of streaming multiprocessors (SMs). Each SM contains a number of SIMD processors, also known as CUDA cores. A multi-threaded program is partitioned into blocks of threads that execute independently from each other. Each SM handles a threaded block at a time, and each SIMD processor carries out a task for each thread.

The CUDA uses heterogeneous programming method, where only the parallel code segments are executed on the device (GPU) while the rest of the program are executed on the host (CPU) in serial. Kernel functions, the code segments to be executed on the device, are called and spawned from the host function. Figure 4.1 shows an example of heterogeneous programming model, where the program alternates between single-threaded host function on the CPU and multi-threaded kernel functions on the GPU. A kernel function is executed as a grid of threaded blocks, and the total number of threads in the function is determined by the dimension of the blocks and grid, as shown in Figure 4.2.

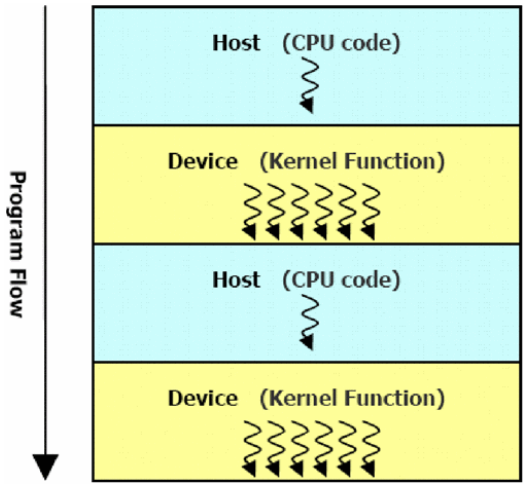


Figure 4.1: Heterogeneous programming model.

4.2 Parallel Integral

In our screening algorithm, the template and all the candidate patches in the reference image are shaped like an octagonal star, which is the superposition of a square area and a diamond area, as shown in Figure 3.5. As the feature calculations of the square area and diamond area of a template or patch are entirely independent from each other, the two subtasks can be performed in parallel. Within each subtask, the computations of different summed area tables for different patch features, such as, patch mean, patch variance and the magnitude of patch gradient, are also mutually independent, and can be carried out in parallel. Therefore, with the aid of GPU, the proposed algorithm can construct all the summed area tables, such as $L_i^{\square}(x, y)$ and $L_i^{\diamond}(x, y), i \in \{1, 2, x, y\}$ as in Eqs. (3.2), (3.14), (3.6) and (3.8), concurrently for different features.

Take the summed area table $L_1^{\square}(x, y)$, which is defined in Eq. 3.2, as an example. With the aid of GPU, table $L_1^{\square}(x, y)$ can be built efficiently by using a two-pass technique. The first pass of the method conducts horizontal cumulative sum on every row of the reference

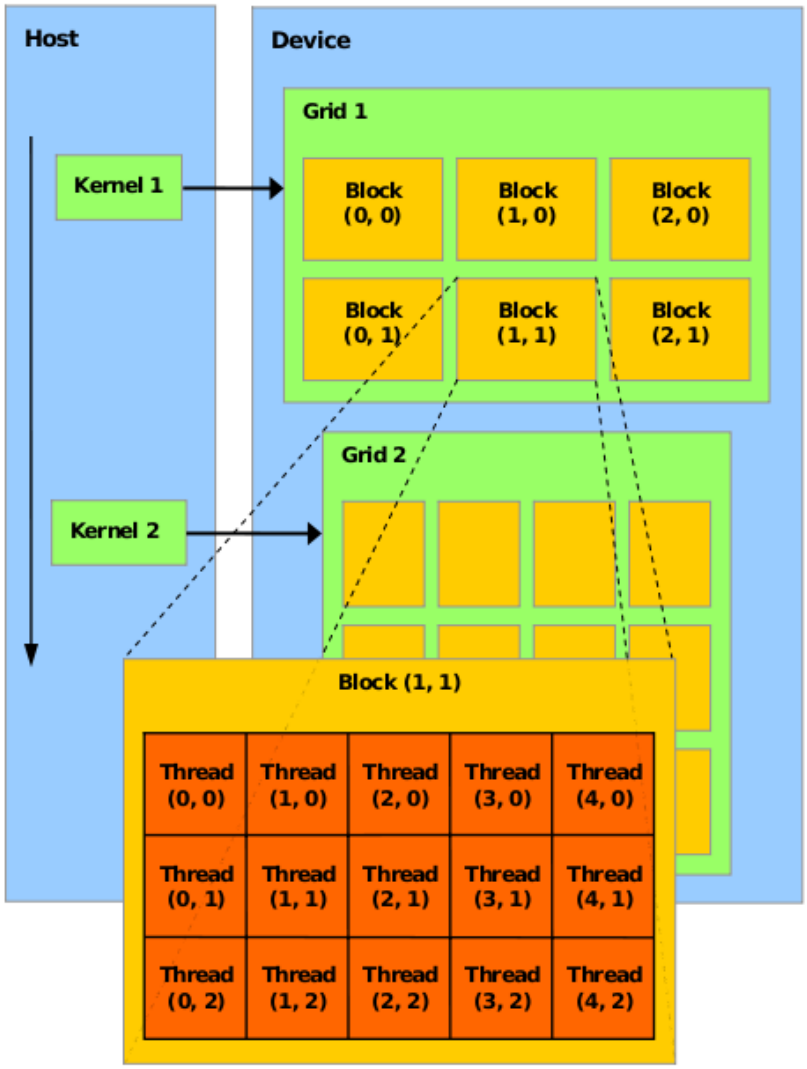


Figure 4.2: Thread, block, and grid arrangement inside kernel functions.

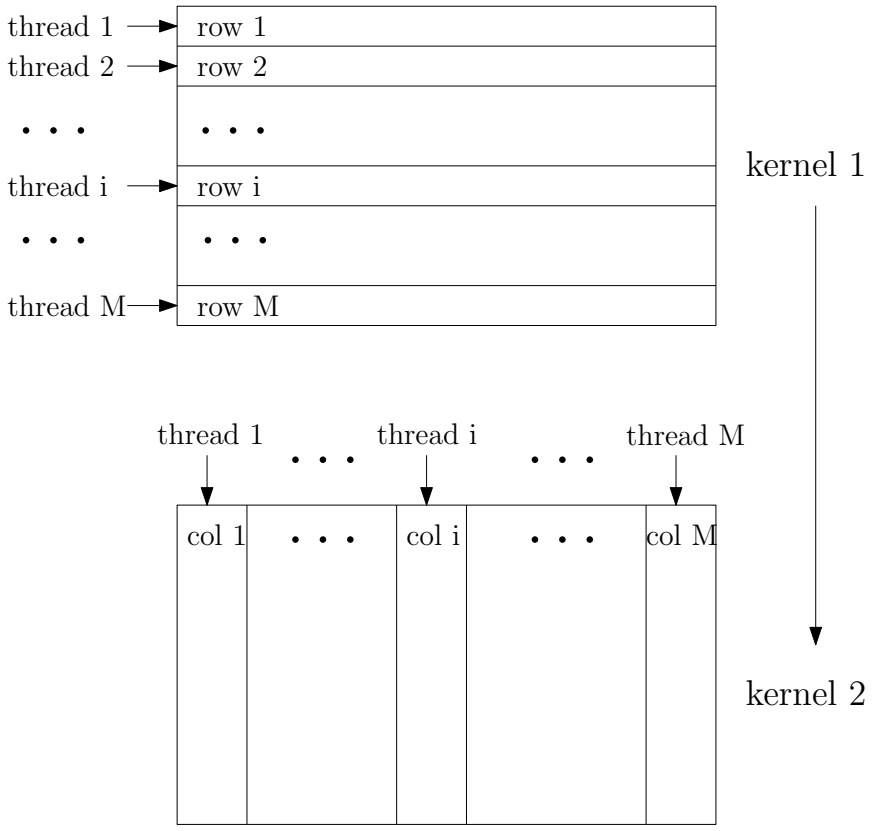


Figure 4.3: Kernel 1 conducts the cumulative sums of M rows of the $M \times M$ reference image while ensuing kernel 2 finishes the cumulative sums of M columns.

image, and then the second pass sums every column cumulatively. Since the cumulative sums are independent from row to row or column to column, it is easy to accelerate the construction process with parallel computing architecture. As shown in Figure 4.3, our parallel algorithm uses a kernel (Kernel 1) with M threads to calculate the cumulative sums of the M rows in the $M \times M$ reference image separately. After the completion of the first kernel, another kernel (Kernel 2) with M threads is invoked to calculate the cumulative sums of the M columns. In each of the two kernels, at most M threads can be carried out concurrently, and each thread takes $O(M)$ time to finish the cumulative sum. Therefore, while the original pixel-wise summed area table establishment requires $O(M^2)$ time, the GPU-based parallel integral only needs $O(M^2/p)$ time, where p is the number of threads running concurrently on the GPU.

In a similar fashion, the two-pass technique for cumulative sums also works effectively for table $L_1^{\diamond}(x, y)$, which is defined in Equation 3.17. As depicted in Figure 4.4, the first pass calculates the sum for each diagonal from top left to bottom right, and the subsequent second pass calculates the diagonal sums from top right to bottom left. For an $M \times M$ reference image, the number of diagonal cumulative sums is $2M-1$ for each pass. Thus, in each kernel, $2M-1$ threads would be assigned and carried out at the same time, achieving overall time complexity $O(M^2/p)$ like the previous case.

Besides tables $L_1^{\square}$ and $L_1^{\diamond}$ for the patch mean, other summed area tables required for calculating the patch variance and patch gradient can also be accelerated using two successive passes.

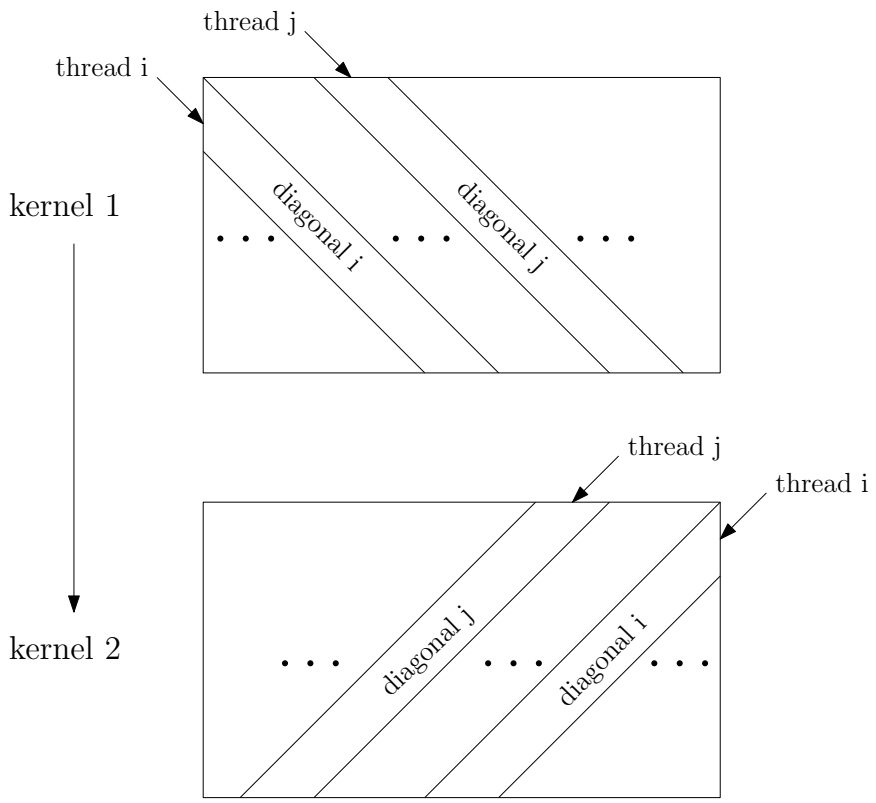


Figure 4.4: Kernel 1 conducts the cumulative sums for $2M-1$ diagonals from top left to bottom right in the reference image while ensuing kernel 2 finishes the cumulative sums of $2M-1$ diagonals from top right to bottom left.

4.3 Parallel Screening

After establishing all the summed area tables, the patch features of an arbitrary patch in the reference image only requires constant time to calculate using the inclusion-exclusion principle. However, as there are $O(M^2)$ candidate patches for an $M \times M$ reference image, it requires $O(M^2)$ time to compute and examine every patch feature for all the patches.

The design of the parallel algorithm for the screening process is straightforward. As shown in Figure 4.5, $(M-n+1)^2$ threads are allocated for all $n \times n$ candidate patches in the reference image. Each thread handles the rotation, scale and illumination invariant template

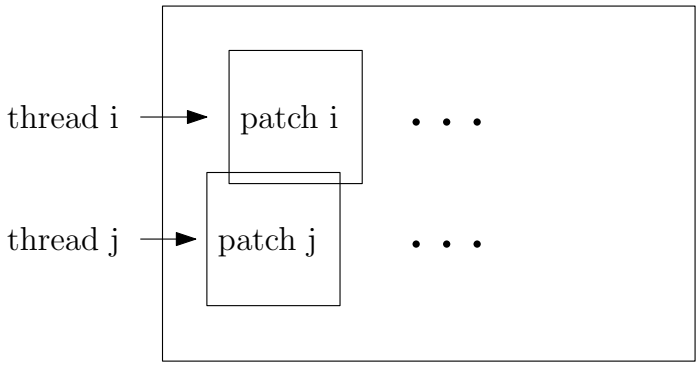


Figure 4.5: Kernel 3: concurrent testing for all the candidate patches.

Global memory	4096 MBytes	Multiprocessors	13
CUDA cores	1664	GPU max clock rate	1.25 GHz
Memory clock rate	3505 MHz	Memory bus width	256 bits
Constant memory	65536 Bytes	Shared memory / block	49152 Bytes
Registers / block	65536	Warp size	32
Maximum threads / multiprocessor	2048	Maximum threads / block	1024

Table 4.1: Device information of NVIDIA GeForce GTX 970.

matching for each patch at the same time, and the thread keeps the qualified candidates for the second-stage processing. The asymptotic time complexity of this parallel algorithm is $O(M^2/p)$, where p is the degree of the parallelism, i.e., the number of threads running concurrently.

4.4 Performance of the Algorithm

In order to evaluate the performance in terms of speed of the proposed parallel algorithms, we conduct experiments of template matching using an NVIDIA GeForce GTX 970, a very common mid-end GPU for personal computer. The device parameters of this GPU are listed in Table 4.1. The CUDA Driver version number is 8.0 and Capability version number is 5.2.

Image size	Template size	time on CPU (s)	time on GPU (s)
640×480	32×32	0.1	0.01
640×480	64×64	0.1	0.01
960×720	32×32	0.2	0.03
960×720	64×64	0.2	0.03
960×720	96×96	0.2	0.03
1280×960	32×32	0.4	0.05
1280×960	64×64	0.4	0.05
1280×960	96×96	0.4	0.05
1280×960	128×128	0.4	0.05

Table 4.2: The performance comparison between CPU-based and GPU-aided implementations on 10 groups of images.

The template matching experiments are conducted with 10 groups of templates and reference images of different sizes. To show the performance differences between the parallel and serial versions of the proposed algorithm, we also tested a serial CPU implementation using the same dataset on a computer with an i7-4770 (3.4GHz) CPU and 8GB memory. The running times of the two parallel and serial implementations are reported in Table 4.2. As shown in the table, GPU-based algorithm is significantly faster than the CPU-based counterpart, demonstrating the feasibility of using parallelism to improve the efficiency of the proposed screening algorithm in real-world applications.

Chapter 5

Experimental Results

To evaluate the performance of the proposed technique, we conduct extensive experiments here. Given a pair of reference image and query template, the pre-processing program marks regions and patches that may match the template and passes the results to a second stage template matching algorithm to pinpoint the exact location of the best match. Three conventional high-precision template matching techniques, SIFT (Lowe, 2004), BBS (Dekel *et al.*, 2015) and Fast-Match (Korman *et al.*, 2013), are tested as the second stage matching algorithm to demonstrate the effectiveness of the proposed pre-processor for different types of template matching techniques. The implementations of the three tested techniques are from their original authors and executed with the default settings. All of the reported experiments in this section are carried on a computer with an Intel i7-4770 (3.4GHz) CPU and 8GB memory, and the presented pre-processing time is also tested from the CPU-based implementation.

Data set	Image size	Template size	Average overlap	Pruning patch	Pruning region	time (s)
I1; T1	640×480	32×32	99.53%	99.71%	81.36%	0.1
I1; T2	640×480	64×64	99.82%	99.72%	74.96%	0.1
I2; T1	960×720	32×32	99.28%	99.79%	88.87%	0.2
I2; T2	960×720	64×64	99.95%	99.84%	81.74%	0.2
I2; T3	960×720	96×96	99.92%	99.88%	74.29%	0.2
I3; T1	1280×960	32×32	99.94%	99.76%	85.87%	0.4
I3; T2	1280×960	64×64	99.98%	99.75%	80.83%	0.4
I3; T3	1280×960	96×96	99.99%	99.74%	77.64%	0.4
I3; T4	1280×960	128×128	99.97%	99.55%	72.63%	0.4

Table 5.1: Statistical results of each data set for image matching.

5.1 Exp. I: Image Matching

In this group of experiments, we evaluate the performance of the proposed pre-processing algorithm for high resolution images. The test images come from the MIT database (Torralba *et al.*, 2009), which covers various scenes like urban streets, indoor and natural environments. We select all the 2250 grayscale images in the database with a resolution no less than 640×480 and divide them into three data sets of images of different sizes with each set containing 750 images. For each image, we extract 2 templates at random locations with random rotations and then scale them to a given size. The scale range is $[0.5, 2]$, i.e., the scale parameters α, β in Algorithm 1 are $0.5, 2$, respectively. To guarantee that these templates can be matched by the conventional template matching techniques in the second stage, we require the standard deviation of the pixel intensities of each template to be above a threshold (Ouyang *et al.*, 2012). In order to make a comprehensive evaluation, we construct 9 data sets with different sizes of templates and images. As shown in Table 5.1, the data sets are denoted as Ii;Tj, for i=1,2,3, j = 1,2,3,4, where i refers to reference image

size, and j refers to template size. For example, I1;T1 and I2;T1 have the same template size 32×32 but different image sizes. Thus, there are 13500 test cases (9 data sets $\times$ 750 images $\times$ 2 templates) in total in this experiment.

With regard to successful screening, we employ a similar definition as in Fast-Match (Korman *et al.*, 2013): the regions preserved by the pre-processing algorithm must overlap at least 90% of the area of the ground truth, otherwise, the screening is considered as failed. By this definition, our proposed scheme never fails in any of the 13500 test cases, and on average, more than 99.8% of the ground truth is preserved after the screening, as presented in Table 5.1. Since the matched patch reported by a conventional template matching algorithm overlaps with the ground truth only by 80% on average in general, our scheme has negligible impact on the accuracy of the second stage algorithm. Figure 5.1 shows some example results of our screening algorithm using the MIT database. The images in the left column are the reference images with two query templates marked in boxes for each image. The remaining regions after screening are shown in the middle and right columns. In general, the more distinct the template is, the more searching space can be ruled out by the proposed algorithm.

Furthermore, our scheme prunes around 80% of the regions on average; and only less than 0.5% of all the candidate patches are marked as possible matches and sent to the second stage algorithm. As a result, the second stage algorithm has a much smaller matching problem to solve and hence requires shorter time. For example, as shown in Figure 5.2, SIFT uses 50% less time by employing the pre-processing algorithm and Fast-Match and BBS also have 30% and 55% reductions, respectively, in time on average. The tests for consuming time of SIFT are conducted on all the data sets, while the timing for Fast-Match and BBS are only tested on partial data sets. Since Fast-Match will be unacceptably slow

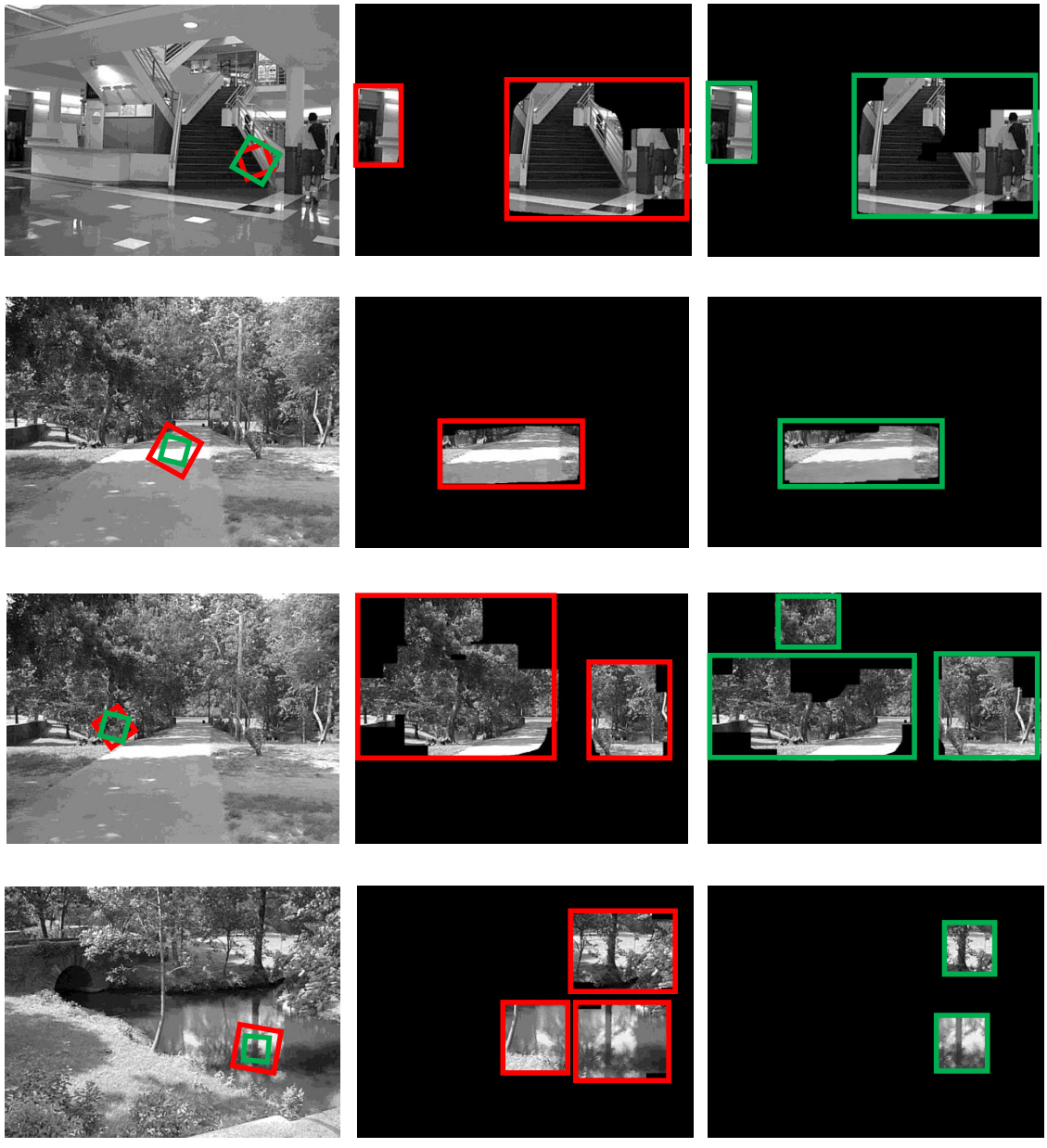
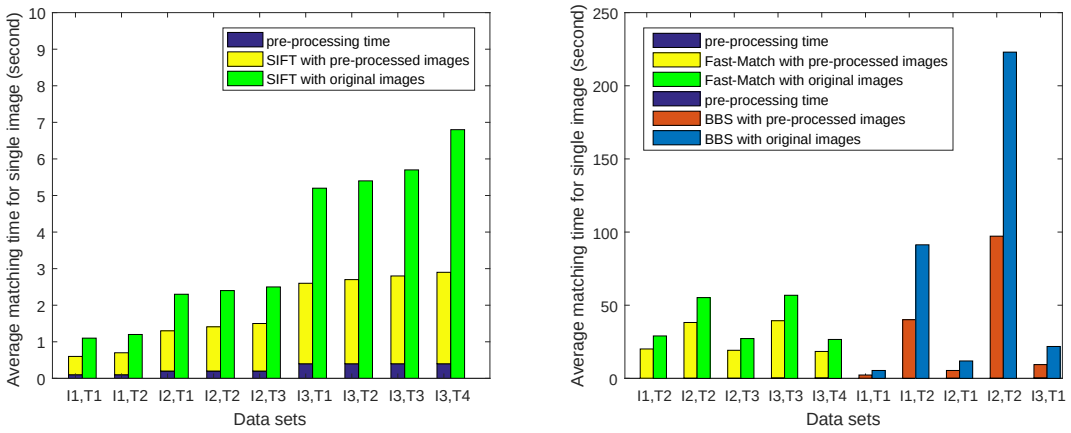


Figure 5.1: Example of screening results for image matching.



(a) SIFT (b) Fast-Match and BBS

Figure 5.2: Comparison of the matching times with or without the proposed pre-processing algorithm.

for small templates in large reference images, such as data set I2;T1, and the running time of BBS also increases rapidly for large templates and large reference images, such as data set I2;T3, the timing tests for those inapplicable data sets are omitted here.

In addition, we also test the effectiveness of the above three matching algorithms with these data sets. SIFT succeeds in over 95% test cases and misses some lightly textured templates. Fast-Match works fine in almost all the test cases. However, BBS cannot handle the large rotation and scaling properly in the synthetic experiment, failing in more than half the test cases. Figure 5.3 shows some matching results of BBS and Fast-Match. The images in left column are randomly selected templates with certain rotation and scaling. The matching results derived by Fast-Match and BBS in original reference images and screened reference images are showed in middle column and right column, respectively, where the red boxes represent the matching results of Fast-Match, and blue boxes pinpoint the best-matched patches from BBS.

Moreover, we can estimate the rotations of each candidate patch relative to template by

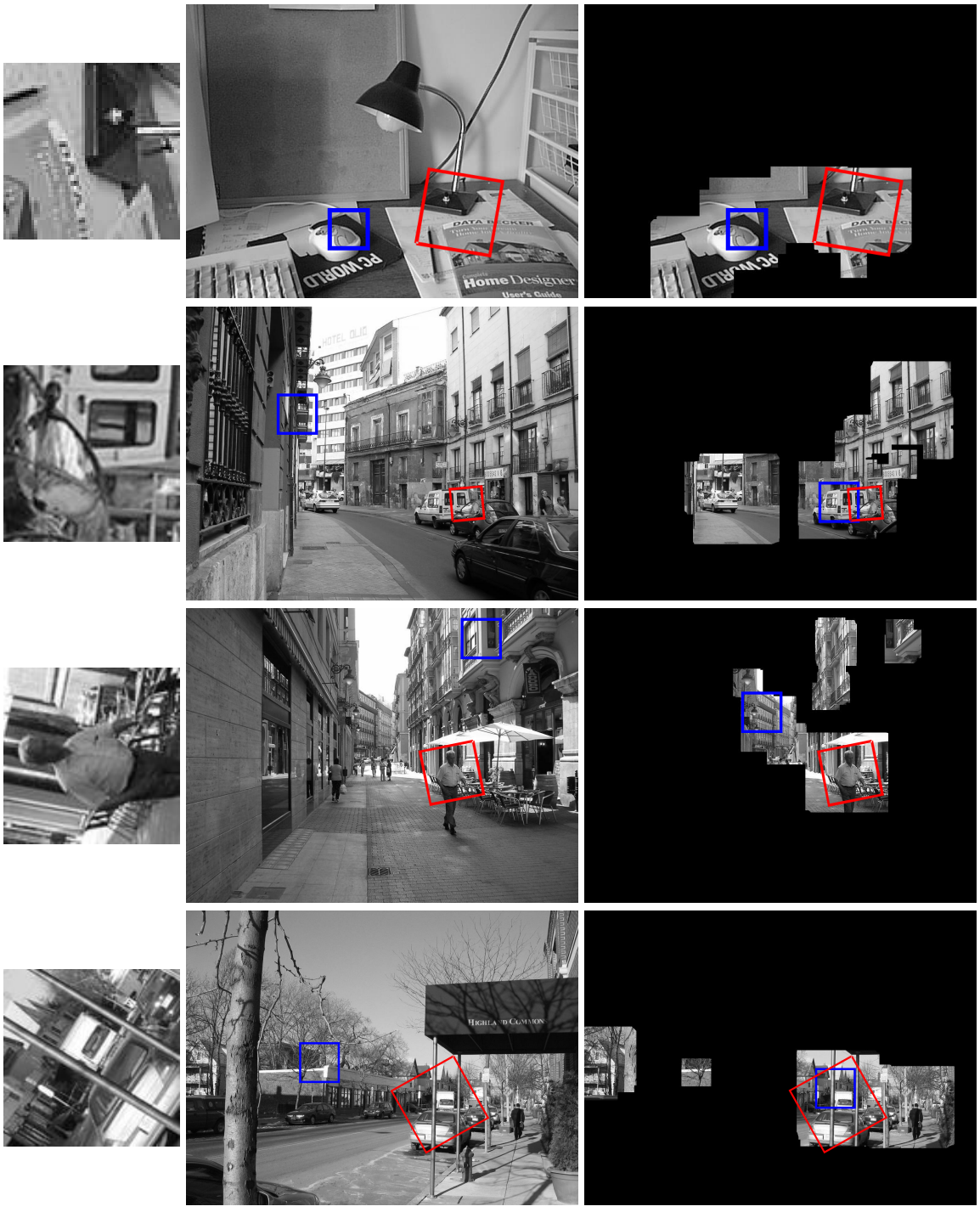
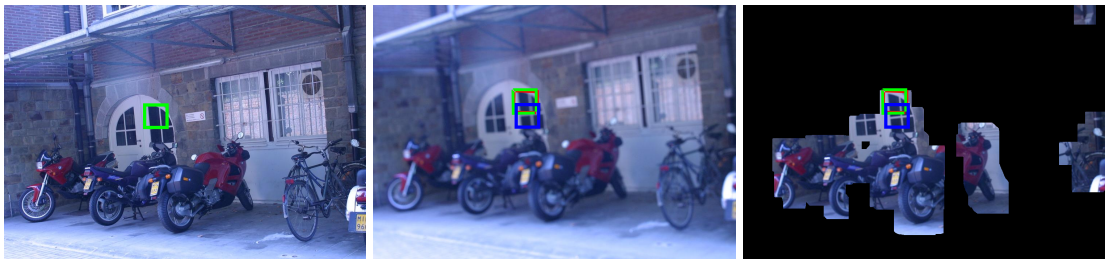


Figure 5.3: Example of matching results before and after screening algorithm.

using its horizontal and vertical gradients and align the orientations of the reference image and the query template. Compared with the ground truth of rotation, the mean absolute error of the our estimates in all the test cases is only 2.20 degree.

5.2 Exp. II: Robustness testing

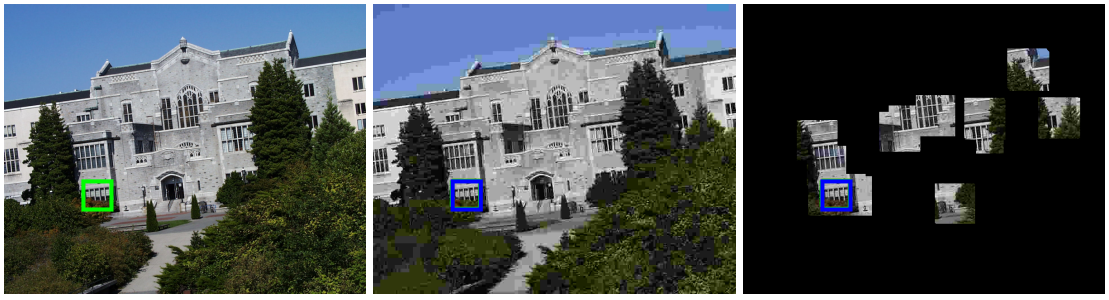
In this experiment, we also use the data set provided in Fast-Match (Korman *et al.*, 2013; Mikolajczyk and Schmid, 2005), to test our algorithm for images with various distortions. This data set consists of 8 sequences, 6 images for each, including image conditions like, zoom, rotation, blur, viewpoint change, brightness change, and JPEG compression. We sample 5 random templates in the first image of each sequence, and conduct screening in the following 5 images respectively. Using the 90% overlap criterion, our screening algorithm successfully preserves the ground truth in all the cases with blur, zoom, rotation and JPEG compression deformations. But for the brightness change, we fail in around 10% testing cases because of the partial illumination changes between the sampled templates and reference images. In view of partial illumination changes, our illumination invariant algorithm designed to deal with uniform illumination changes does not work. Additionally, the success ratio falls to around 60% for test cases with significant viewpoint change. However, overall the screening has negligible impact to the results of the conventional algorithms, as they usually cannot find the best match as well for those difficult cases. As demonstrated in Figure 5.4 are some sample results for images with different distortions, where the green boxes in the left column mark the templates and the green, red, blue boxes in the middle and right columns mark the ground truth, the results by Fast-Match and the results by BBS, respectively.



(a) Blur

(b) Brightness change

(c) Viewpoint change



(d) JPEG compression

Figure 5.4: Example of screening results for images with distortions.

Algorithms	Results	Without screening	With screening
Fast-Match	Time per image (s)	36.2s	25.3s
	Average overlap	41%	41%
	Success ratio	45/105	46/105
BBS	Time per image (s)	16.7s	9.4s
	Average overlap	62%	61%
	Success ratio	75/105	75/105
SIFT	Time per image (s)	1.8s	1.8s
	Average overlap	N/A	N/A
	Success ratio	10/105	10/105

Table 5.2: Statistical results of each tested algorithm for video tracking.

5.3 Exp. III: Video Tracking

This group of experiments evaluates our algorithm for video tracking applications. The test set is generated from 35 color video clips provided in the Visual Tracker Benchmark (Wu *et al.*, 2013). From each video, we randomly pick three pairs of frames that are 20 frames apart and use the annotated object in the first frame as template and the second frame as reference image for each frame pair. This test set is quite challenging, because the objects of interest typically undergo some non-rigid deformations and may also be partially occluded after 20 frames.

The average running time of our pre-processing algorithm on this data set is only 0.05s, as the resolution of the videos is relatively low (480×320). On average, our algorithm prunes 88% of the patches and 60% of the regions, and in 95 out of the 105 cases, it preserves more than 90% of the ground truth after screening. Although in some cases, the screening is not successful (>90% overlap), our proposed algorithm does not adversely affect the success ratio of the tested conventional algorithm at all as shown in Table 5.2. Interestingly, the success ratio becomes slightly higher for Fast-Match when the screening

Figure 5.5: Example of screening results for video tracking.

algorithm is applied. This is because a conventional algorithm may match to a wrong patch in some cases, but the wrong patch can be recognized and removed by our algorithm beforehand. Some sample results of the video tracking data set are shown in Figure 5.5. The results of each algorithm are marked in the same way as Figure 5.4.

Chapter 6

Conclusion

This paper presents a generic template matching pre-processor for expediting conventional template matching techniques. The proposed pre-processing algorithm can handle rotation, scale and illumination change of the reference image effectively as demonstrated by the experimental results in Section 5.1. To make the proposed algorithm robust against rotation, we shape the template as an octagonal star. Other than being approximate to a circle, which is ideal for rotation invariant matching, the advantage of using the octagonal star is that each feature, such as mean, variance and magnitude, of the template can be calculated efficiently using the inclusion-exclusion principle. The scale invariance is achieved by enumerating all possible scales like many previous techniques. However, by utilizing the statistical fact that matched patches likely have matched central areas, the proposed algorithm only needs to examine a very small number of scales in the reference image, significantly reducing the computational cost of scale invariant matching. Finally, the proposed algorithm alleviates the effects of illumination change by employing patch features normalized by mean intensity. As shown in Section 5.2, not only is the proposed algorithm robust against rotation, scale and illumination changes, but it also works well with some other types of image

degradation like blur and JPEG compression.

Designed with the computational efficiency as one of the top priorities, the proposed pre-processor is significantly faster than all mainstream rotation and scale invariant image matching algorithms as discussed in Section 5.3. Moreover, the proposed technique can take full advantage of modern parallel computing architecture like GPU to further improve its throughput. As a result, the proposed technique can expedite various types of image matching techniques, such as feature matching algorithm SIFT, affine invariant template matching algorithm Fast-Match, robust template matching algorithm BBS, with almost no penalties in matching accuracy.

For future study, we plan to improve and extend our fast screening algorithm in the following aspects. First, we will add support of colour image in the algorithm. To simplify the comparisons with previous techniques, only the intensity channel of an input image is utilized by the current version of the algorithm as discussed in the thesis, although in most applications, the query template and reference image are in colour. The extra colour information is likely to be helpful in improving the matching accuracy and increasing the pruning ratio of the proposed algorithm at the expense of the screening speed. The key is to balance the pruning ratio and computational cost so that the two stage approach achieves the maximum overall efficiency without affecting the matching accuracy, as discussed in the previous sections.

Second, as the sampled area of a template for feature extraction is an octagonal star, we require the query template to be a square rather than any arbitrary rectangle to match the outline of the octagonal star. While using square template is very common in the literature, it does pose an unnecessarily strict restriction for real world applications. A quick solution to this problem is to split a rectangle template into several square parts and

store the features of all parts into the same membership array. Then the screening algorithm can check whether a patch matches some part of the query template. The computational complexity of this simple scheme is still $O(M^2 + n^2)$, same as the square template case. However, although this method does not reduce the matching accuracy, it can adversely affect the pruning ratio, since each part of the template is now matched individually. We will investigate this issue in the future and find an efficient and effective method that utilizes not only the features of each part of the template but also the relative positions of the different parts.

Lastly, we would like to extend our work to template matching with multiple observations. Most existing template matching algorithms use only one image of the target object as the query template, but in majority of the applications, there are often a large number of observations of the object readily available as template. For instance, in video tracking, the target object has already been marked in previous frames when the current frame is processed; each of these previous frames is a valid observation for tracking the object in the current frame. Another example is 3D template matching where the object is given as a 3D model rather than a single image. In this case, the target object can have multiple vastly different renderings from different angles and with different lightings as template. Fully utilizing these existing observations should greatly improve the robustness and accuracy of a matching algorithm. Our current framework of template matching pre-processor is flexible and should have no problem in incorporating multiple observations in the work flow. Similar to the aforementioned idea for supporting rectangular template, we can aggregate features of multiple templates into the same membership array and then compare a patch against all the templates simultaneously. This approach only increases the complexity of building a feature set but does not affect the efficiency of the most time consuming part

of the algorithm — scanning through the reference image. However, in order to make this approach effective, the quantization factor of each feature must be carefully designed. Due to the large number of valid features in the feature set from the additional templates, the likelihood of falsely marking an unmatched patch as a potential match is much higher than the previous single template case.

Bibliography

Agrawal, M., Konolige, K., and Blas, M. R. (2008). Censure: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, pages 102–115. Springer.

Ashbrook, A., Thacker, N. A., Rockett, P. I., and Brown, C. (1995). Robust recognition of scaled shapes using pairwise geometric histograms. In *BMVC*, volume 95, pages 503–512.

Babaud, J., Witkin, A. P., Baudin, M., and Duda, R. O. (1986). Uniqueness of the gaussian kernel for scale-space filtering. *IEEE transactions on pattern analysis and machine intelligence*, (1), 26–33.

Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE transactions on pattern analysis and machine intelligence*, **24**(4), 509–522.

Brown, L. G. (1992). A survey of image registration techniques. *ACM computing surveys (CSUR)*, **24**(4), 325–376.

Calonder, M., Lepetit, V., Fua, P., Konolige, K., Bowman, J., and Mihelich, P. (2009). Compact signatures for high-speed interest point description and matching. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 357–364. IEEE.

Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. *Computer Vision–ECCV 2010*, pages 778–792.

Choi, M.-S. and Kim, W.-Y. (2002). A novel two stage template matching method for rotation and illumination invariance. *Pattern recognition*, **35**(1), 119–129.

Crow, F. C. (1984). Summed-area tables for texture mapping. *ACM SIGGRAPH computer graphics*, **18**(3), 207–212.

Dekel, T., Oron, S., Rubinstein, M., Avidan, S., and Freeman, W. T. (2015). Best-buddies similarity for robust template matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2021–2029.

Deledalle, C.-A., Salmon, J., Dalalyan, A. S., *et al.* (2011). Image denoising with patch based pca: local versus global. In *BMVC*, volume 81, pages 425–455.

Fischler, M. A. and Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, **24**(6), 381–395.

Gharavi-Alkhansari, M. (2001). A fast globally optimal algorithm for template matching using low-resolution pruning. *IEEE Transactions on Image Processing*, **10**(4), 526–533.

Hinterstoisser, S., Cagniart, C., Ilic, S., Sturm, P., Navab, N., Fua, P., and Lepetit, V. (2012). Gradient response maps for real-time detection of textureless objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **34**(5), 876–888.

Hua, G., Brown, M., and Winder, S. (2007). Discriminant embedding for local image descriptors. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Ke, Y. and Sukthankar, R. (2004). Pca-sift: A more distinctive representation for local image descriptors. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–506. IEEE.

Kekre, H. and Gharge, S. (2010). Image segmentation using extended edge operator for mammographic images. *International journal on computer science and Engineering*, **2**(4), 1086–1091.

Kim, H. Y. (2010). Rotation-discriminating template matching based on fourier coefficients of radial projections with robustness to scaling and partial occlusion. *Pattern Recognition*, **43**(3), 859–872.

Korman, S., Reichman, D., Tsur, G., and Avidan, S. (2013). Fast-match: Fast affine template matching. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2331–2338.

Lewis, J. (1995). Fast normalized cross-correlation. In *Vision interface*, volume 10, pages 120–123.

Lienhart, R. and Maydt, J. (2002). An extended set of haar-like features for rapid object detection. In *Image Processing. 2002. Proceedings. 2002 International Conference on*, volume 1, pages I–900. IEEE.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Lin, Y.-H. and Chen, C.-H. (2008). Template matching using the parametric template vector with translation, rotation and scale invariance. *Pattern Recognition*, **41**(7), 2413–2421.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, **60**(2), 91–110.

Messom, C. and Barczak, A. (2006). Fast and efficient rotated haar-like features using rotated integral images. In *Australian Conference on Robotics and Automation*, pages 1–6.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE transactions on pattern analysis and machine intelligence*, **27**(10), 1615–1630.

Morel, J.-M. and Yu, G. (2009). Asift: A new framework for fully affine invariant image comparison. *SIAM Journal on Imaging Sciences*, **2**(2), 438–469.

Oron, S., Dekel, T., Xue, T., Freeman, W. T., and Avidan, S. (2016). Best-buddies similarity-robust template matching using mutual nearest neighbors. *arXiv preprint arXiv:1609.01571*.

Ouyang, W., Tombari, F., Mattoccia, S., Di Stefano, L., and Cham, W.-K. (2012). Performance evaluation of full search equivalent pattern matching algorithms. *IEEE transactions on pattern analysis and machine intelligence*, **34**(1), 127–143.

Pele, O. and Werman, M. (2007). Accelerating pattern matching or how much can you slide? In *Asian Conference on Computer Vision*, pages 435–446. Springer.

Schaffalitzky, F. and Zisserman, A. (2002). Multi-view matching for unordered image sets, or how do i organize my holiday snaps?. *Computer VisionECCV 2002*, pages 414–431.

Schweitzer, M., Jaspers, H., Kubertschak, T., and Wuensche, H.-J. (2013). Parallelized 45 degrees rotated image integration. In *Image Processing (ICIP), 2013 20th IEEE International Conference on*, pages 810–814. IEEE.

Shakhnarovich, G. (2005). *Learning task-specific similarity*. Ph.D. thesis, Massachusetts Institute of Technology.

Storer, J. (2001). *An Introduction to Data Structures and Algorithms*. Springer Science & Business Media.

Torralba, A., Murphy, K., and Freeman, W. (2009). The mit csail database of objects and scenes.

Tsai, D.-M. and Chiang, C.-H. (2002). Rotation-invariant pattern matching using wavelet decomposition. *Pattern Recognition Letters*, **23**(1), 191–201.

Tuytelaars, T. and Schmid, C. (2007). Vector quantizing feature space with a regular lattice. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8. IEEE.

Ullah, F. and Kaneko, S. (2004). Using orientation codes for rotation-invariant template matching. *Pattern recognition*, **37**(2), 201–209.

Van Gool, L., Moons, T., and Ungureanu, D. (1996). Affine/photometric invariants for planar intensity patterns. In *European Conference on Computer Vision*, pages 642–651. Springer.

Viola, P. and Jones, M. J. (2004). Robust real-time face detection. *International journal of computer vision*, **57**(2), 137–154.

Wakahara, T. and Yamashita, Y. (2014). Gpt correlation for 2d projection transformation invariant template matching. In *Pattern Recognition (ICPR), 2014 22nd International Conference on*, pages 3810–3815. IEEE.

Winder, S., Hua, G., and Brown, M. (2009). Picking the best daisy. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 178–185. IEEE.

Wu, Y., Lim, J., and Yang, M.-H. (2013). Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2411–2418.